Constructing Zero-deficiency Parallel Prefix Circuits of Minimum Depth

HAIKUN ZHU, CHUNG-KUAN CHENG and RONALD GRAHAM
University of California, San Diego

A parallel prefix circuit has n inputs $x_1, x_2, \ldots, x_n$, and computes the n outputs $y_i = x_i \bullet x_{i-1} \bullet \cdots \bullet x_1, 1 \leq i \leq n$, in parallel, where $\bullet$ is an arbitrary binary associative operator. Snir proved that the depth t and size s of any parallel prefix circuit satisfy the inequality $t + s \geq 2n - 2$. Hence, a parallel prefix circuit is said to be of zero-deficiency if equality holds. In this paper, we provide a different proof for Snir's theorem by capturing the structural information of zero-deficiency prefix circuits. Following our proof, we propose a new kind of zero-deficiency prefix circuit $Z(d)$ by constructing a prefix circuit as wide as possible for a given depth d. It is proved that the $Z(d)$ circuit has the minimal depth among all possible zero-deficiency prefix circuits.

Categories and Subject Descriptors: B.6.1 [**Logic Design**]: Design Styles—*parallel circuits*

General Terms: Algorithms, Design, Theory

Additional Key Words and Phrases: Zero-deficiency, parallel prefix circuits, depth-size tradeoff

1. INTRODUCTION

1.1 Problem Definition

The prefix problem, which mostly gains research attention with the emergence of parallel computing, is actually the abstraction of many practical applications such as binary addition, radix sort, linear recurrences solving, polynomial evaluation, etc. [Lakshmivarahan and Dhall 1994]. Formally, the prefix problem is defined as follows:

Definition 1.1. **[Prefix Problem]** Given n inputs $x_1, x_2, \ldots, x_n$ and an arbitrary binary associative operator $\bullet$, compute the prefix results $Y_i = x_i \bullet x_{i-1} \bullet \cdots \bullet x_1$ for $1 \leq i \leq n$.

Since the binary operator $\bullet$ is associative, it is convenient to introduce the *partial prefix result* $y_{[i:j]}$using the following recurrence formula:

$$y_{[i:i]} = x_i, 1 \leq i \leq n \quad (1)$$

$$y_{[i:j]} = y_{[i:k]} \bullet y_{[k-1:j]}, 1 \leq j < k \leq i \leq n \quad (2)$$

This work was supported in part under grants from NSF project number MIP-9987678, the California MICRO program and SRC support.
Authors's addresses: CSE Department, University of California, San Diego, La Jolla, CA 92093-0114; email:{hazhu,kuan,rgraham}@cs.ucsd.edu.

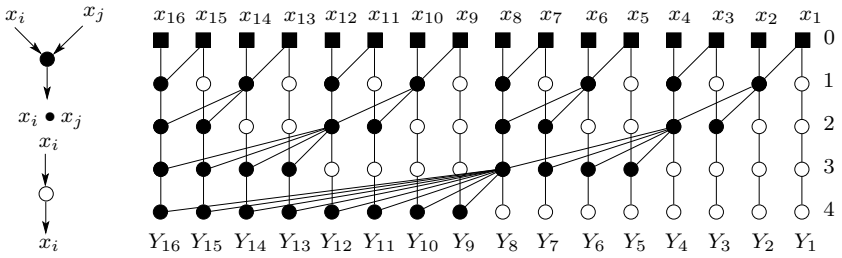

Fig. 1. An example of the parallel prefix circuit: Sklansky's structure.

Using the above representation, Y_i is nothing but a special kind of partial prefix result:

$$Y_i = y_{[i:1]} \tag{3}$$

Nevertheless, the Y_i are usually referred to as *final prefix results* so as to be distinguishable from other intermediate partial prefix results.

Note that the associative operator $\bullet$ in general may not be commutative. Thus in computing Y_i it is imperative to keep the order of the primary inputs x_i as it is defined.

Typically, the prefix problem is visualized by a directed acyclic graph, referred to as the *parallel prefix circuit*, which takes in the n inputs $x_i (1 \leq i \leq n)$ and produces the n final prefix results $Y_i (1 \leq i \leq n)$ in parallel. The computation nodes in the prefix circuit are arranged in levels representing the timing of the results. Each computation node has two inputs, which can be either primary inputs or partial prefix results from previous levels, and produces either another partial or a final prefix result. All the computation nodes that generate $y_{[i:j]}$ with the same i are in the same column. As an example, Figure 1 shows a 16-input Sklansky prefix circuit [Sklansky 1960] which produces all the prefix results in 4 levels, level 0 being the primary inputs. The primary inputs are represented by solid squares while the computation nodes are represented by solid black nodes. The white nodes are simply placeholders which do nothing but pass through the data. In the rest of this paper, we always mean computation nodes when we say "nodes", unless otherwise stated.

It is then natural to define the size and depth of the prefix circuits as follows:

Definition 1.2. Denote a prefix circuit of n inputs as $G(n)$. In the absence of confusion, we say the width of $G(n)$ is n. The size of $G(n)$, denoted by $s_{G(n)}$, is the number of computation nodes in $G(n)$. Similarly, the depth of $G(n)$, denoted by $d_{G(n)}$, is the number of levels in $G(n)$.

The main problem in designing prefix circuits has been identifying the exact tradeoff between the size and the depth. Snir [1986] proved that $s_{G(n)} + d_{G(n)} \geq 2n - 2$ holds for arbitrary prefix circuits, based on a concept he introduced called zero-deficiency:

Definition 1.3. The deficiency of a prefix circuit $G(n)$ is defined as

$$def(G(n)) = s_{G(n)} + d_{G(n)} - (2n - 2) \tag{4}$$

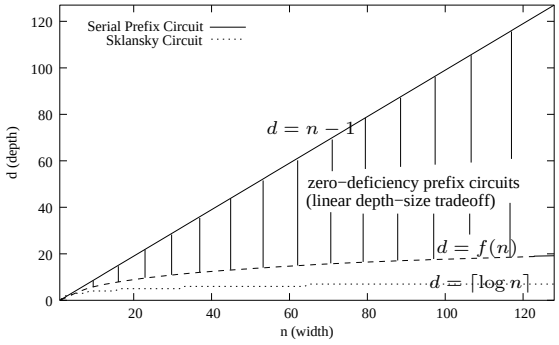

Fig. 2. Depth-Size tradeoffs of the parallel prefix circuits.

$G(n)$ is said to be of **zero-deficiency** if $def(G(n)) = 0$.

Snir's theorem indicates that the solution space for parallel prefix circuits should look like Figure 2. For a given width n, the maximum depth of the prefix circuits is $n - 1$ (serial prefix circuit) while the minimum depth is $\lceil \log n \rceil$ (Sklansky's circuit [Sklansky 1960]). For loose depth constraints we can observe a linear tradeoff between the depth and the size which is exhibited by the zero-deficiency prefix circuits. However, if the depth constraint is too tight, the size of the prefix circuits will grow dramatically and zero-deficiency prefix circuits no longer exist. A direct example of this is the Sklansky's structure in Figure 1, which has depth k and size $k2^{k-1}$ for width 2^k. Nevertheless, even when zero-deficiency is not achievable, there is certainly a lower bound for the size under the given depth constraint. Hence, we introduce the concept of *depth-size optimal*:

Definition 1.4. A parallel prefix circuit is defined as depth-size optimal if it achieves the minimum size for the given depth constraint.

We shall stress the difference between zero-deficiency and depth-size optimal. A zero-deficiency prefix circuit must be depth-size optimal, but the converse may not hold. In Figure 2, the region between curves $d = f(n)$ and $d = \lceil \log n \rceil$ is where zero-deficiency prefix circuits do not exist, but depth-size optimal prefix circuits do. It remains an open question to find the zero-deficiency prefix circuits of minimum depth, which is the subject of this paper.

1.2 Previous Work

Various zero-deficiency prefix circuits were proposed in the past. In [1986], Snir gave a class of zero-deficiency prefix circuits by concatenating a Brent-Kung circuit with a serial prefix circuit. His design requires that $d_{G(n)} \geq 2\lceil \log n \rceil - 2$. Following the same idea, Lin and Shih were able to improve the minimum depth to as small as $2\lceil \log n \rceil - 5$ by fine tuning the widths of the Brent-Kung circuits and the serial circuits [Lin and Shih 1999]. Another kind of zero-deficiency circuit of small depth is the $LYD(n)$ circuit which comprises a Brent-Kung circuit, a delay-balanced two-level serial circuit, and two one-level serial circuits [Lakshmivarahan et al. 1987]. In [1996], Zimmermann proposed a deterministic algorithm for prefix circuit optimization using depth-controlled compression and expansion. Although his approach in

many cases produces depth-size optimal or near optimal prefix circuits, optimality of his results is not guaranteed.

Historically, there are also some other prefix circuit designs such as those in [Ladner and Fischer 1980], [Fich 1983] and [Bilgory and Gajski 1986]. However, these prefix circuit structures have non-zero deficiency, and are therefore inferior to the zero-deficiency prefix circuits considered in this paper.

In this paper, we propose a new kind of zero-deficiency prefix circuit called $Z(d)$ which provably has the minimum depth for a given width. We managed to do so from an alternative point of view, that is, by constructing a zero-deficiency prefix circuit of maximum width for a given depth. Our new philosophy leads to a systematic way of prefix circuit construction, as opposed to previous ad-hoc approaches.

The remainder of the paper is organized as follows. In Section 2, we first give a revised proof for Snir's lower bound theorem $s_{G(n)} + d_{G(n)} \geq 2n - 2$, which is enlightened by similar ideas in [Fich 1983] and [Lakshmivarahan and Dhall 1994], and then discuss the properties of the zero-deficiency circuits. Our main contribution lies in Section 3, where a new class of zero-deficiency prefix circuits $Z(d)$ is proposed. We will prove that $Z(d)$ has the minimum depth among all possible zero-deficiency prefix circuits. In Section 4, we extend $Z(d)$ circuit to consider arbitrary width and depth constraints. Section 5 offers a few comments on the proposed structure and compares it with some other work. Section 6 concludes the paper.

2. PROPERTIES OF THE ZERO-DEFICIENCY PREFIX CIRCUITS

Snir's original proof for $s_{G(n)} + d_{G(n)} \geq 2n - 2$ is by mathematical induction and does not reveal the structural information of zero-deficiency circuits. We notice that there are some nice ideas in [Fich 1983] and [Lakshmivarahan and Dhall 1994] which can be used to devise an elegant proof for Snir's theorem. In this section, we polish these ideas and prove an enhanced version of Snir's lower bound theorem.

THEOREM 2.1. *For an n-input prefix circuit $G(n)$, denote the depth of its last prefix output Y_n as $d^M_{G(n)}$. Then*

$$s_{G(n)} + d_{G(n)} \geq s_{G(n)} + d^M_{G(n)} \geq 2n - 2 \tag{5}$$

PROOF. Let us consider the node that generates the last final prefix result Y_n, and use Figure 3 as an illustration. This output node has to cover all the primary inputs, hence the structure that generates Y_n must be an upside-down tree with all the primary inputs being its leaves. Let us name it the **primal fan-in tree**. Note that some nodes in the primal fan-in tree may have fan-outs to prefix outputs other than Y_n, but those fan-outs and their succeeding nodes do not constitute the primal fan-in tree. Put another way, one can identify the primal fan-in tree by starting from the node Y_n and tracing its inputs all the way back to the primary inputs. Remember that the binary operator is not commutative in general, and the order of the primary inputs is fixed. Thus the primal fan-in tree is essentially a binary alphabetical tree. To illustrate, the edges of the primal fan-in tree in Figure 3 are highlighted using heavy lines. The size of the primal fan-in tree is exactly $n - 1$, and its depth is $d^M_{G(n)}$.

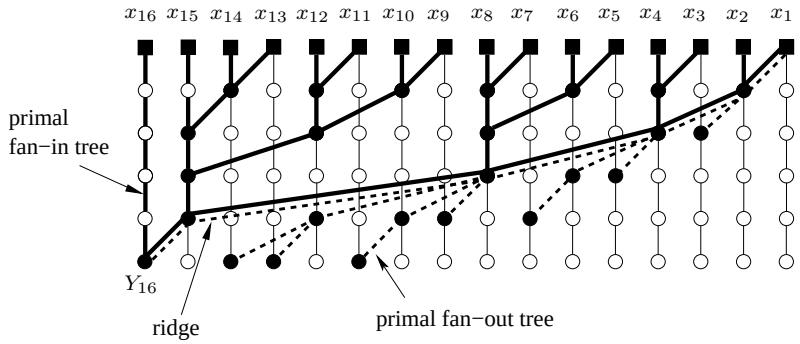

Fig. 3. The primal fan-in tree, primal fan-out tree and ridge of a parallel prefix circuit.

On the other hand, the first primary input x_1 has to be fed into every final prefix output, either directly or indirectly. Consequently, there is another tree rooted at x_1 with the final prefix outputs Y_i $(1 \leq i \leq n-1)$ as its leaves. Let us call it the **primal fan-out tree**. The primal fan-out tree may not necessarily be a binary tree, since one node may fan out to more than two nodes of next level. Nevertheless, it is clear that the primal fan-out tree has at least $n-1$ nodes, which are just the $n-1$ final prefix outputs Y_i $(1 \leq i \leq n-1)$. In Figure 3, the edges of the primal fan-out tree are shown by heavy dotted lines.

Notice that the primal fan-in tree and the primal fan-out tree actually overlap on the path from the first primary input x_1 to the last prefix output Y_n, which we call the **ridge** of the prefix circuit. There are at most $d^M_{G(n)}$ nodes on the ridge. Up to now we have had $(n-1)+(n-1)-d^M_{G(n)} = 2n-2-d^M_{G(n)}$ nodes in the prefix circuit. Since the primal fan-in tree and primal fan-out tree shall exist in every prefix circuit, this is indeed the minimum number of nodes for a general prefix circuit. Hence we have

$$s_{G(n)} + d_{G(n)} \geq s_{G(n)} + d^M_{G(n)} \geq 2n-2$$

□

In fact, each final prefix output Y_i $(1 \leq i \leq n)$ is generated by a binary alphabetical tree. Only the one generates Y_n is called primal because it covers all the primary inputs. Likewise, from each primary input x_i $(1 \leq i \leq n)$ sprouts a fan-out tree to Y_j $(i \leq j \leq n)$. Only the one rooted at x_1 is called primal because it feeds all of the final prefix outputs.

It can be inferred from the above proof that a zero-deficiency prefix circuit has no nodes other than those on its primal fan-in tree and primal fan-out tree. For example, the Sklansky's prefix circuit [Sklansky 1960] in Figure 4(a) has 6 nodes in addition to its primal fan-in tree and primal fan-out tree, thereby it is not a zero-deficiency circuit. Furthermore, for a zero-deficiency prefix circuit the last output Y_n must also be the latest output. For instance, the Brent-Kung circuit [Brent and Kung 1982] shown in Figure 4(b) is not a zero-deficiency circuit because Y_{16} is two levels ahead of Y_{15}, although Brent-Kung circuit consists of only the primal fan-in tree and primal fan-out tree.

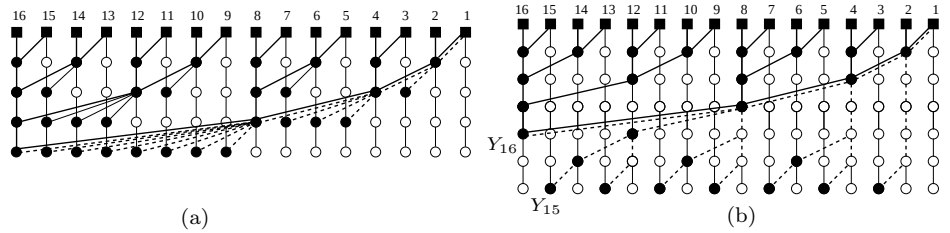

Fig. 4. (a) Sklansky's prefix circuit; and (b) Brent-Kung circuit; The edges of the primal fan-in tree are emphasized by heavy lines, while the edges of the primal fan-out tree are emphasized by heavy dotted lines.

In summary, Definition 2.2 gives a formal treatment of the concepts involved in the proof of Theorem 2.1, while Corollary 2.3 gives the necessary and sufficient conditions for a prefix circuit to be of zero-deficiency.

Definition 2.2. For a prefix circuit $G(n)$, the binary alphabetical tree that generates the last final prefix output Y_n is called the **primal fan-in tree** of $G(n)$. The broadcasting tree that propagates the first primary input x_1 to the final prefix outputs Y_i $(1 \leq i \leq n)$ is defined as the **primal fan-out tree** of $G(n)$. The common part of the primal fan-in tree and the primal fan-out tree, that is, the path from the first primary input to the last final prefix output, is defined as the **ridge** of $G(n)$.

COROLLARY 2.3. *A prefix circuit $G(n)$ of depth d is of zero-deficiency if and only if*

(1) $G(n)$ comprises only the primal fan-in tree and the primal fan-out tree, which overlap on its ridge;

(2) $d^M_{G(n)} = d$, i.e., the last prefix output Y_n is also the latest output;

(3) The ridge has d nodes, one node per level.

3. A NEW CLASS OF ZERO-DEFICIENCY PREFIX CIRCUITS

In this section, we propose a new class of zero-deficiency prefix circuits, called $Z(d)$, which have the minimum depth among all zero-deficiency prefix circuits.

3.1 Building the components of Z(d) circuit: $T^k(t)$ trees and $A^k(t)$ trees

We will first construct a class of parameterized trees called $T^k(t)$ trees which will be used to form the primal fan-in tree of the $Z(d)$ circuit. We then define the $A^k(t)$ trees which will be used to form the primal fan-out tree of $Z(d)$. A $Z(d)$ circuit is constructed by assembling $T^k(t)$ trees and $A^k(t)$ trees together.

The $T^k(t)$ trees follow a recursive definition as described in Figure 5. Note that Figure 5(a), (b) and (c) together define $T^k(t)$ over $1 \leq k \leq t$. A few more observations can be made by investigating the graphical definition carefully:

—$T^1(t)$ in Figure 5(a) are just the serial prefix circuits.

—The width of a $T^k(k)$ tree is 2^k for any $k \geq 1$.

—A $T^k(t)$ tree is a binary tree of depth t for $1 \leq k \leq t$.

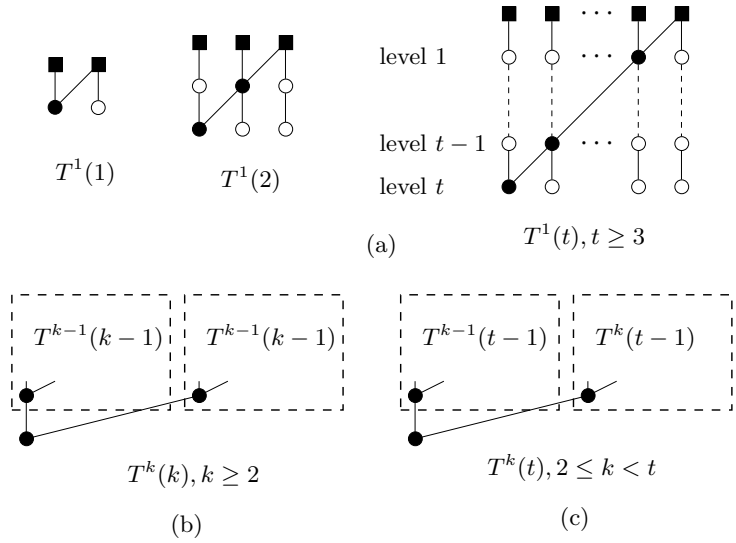

Fig. 5. The recursive definition of the $T^k(t)$ trees: (a) $T^1(t)$, $t \geq 1$; (b) $T^k(k)$, $k \geq 2$; (c) $T^k(t)$, $2 \leq k < t$.

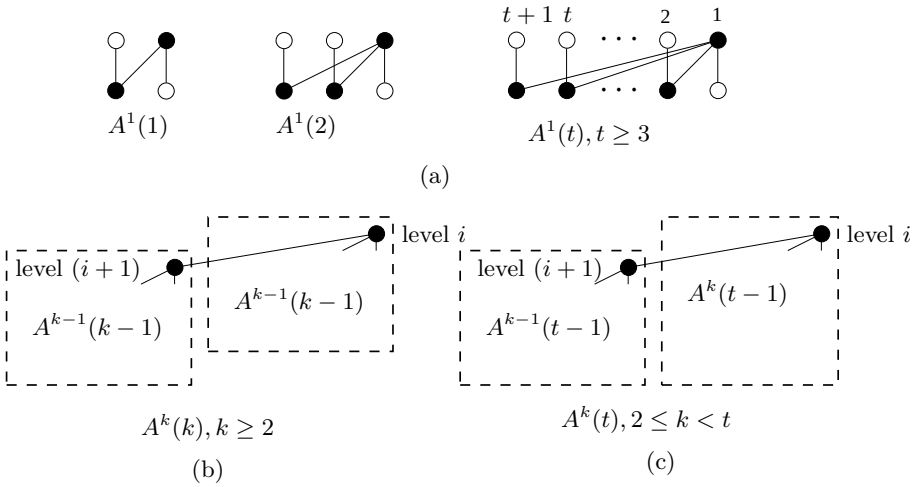

Fig. 6. The recursive definition of the $A^k(t)$ trees: (a) $A^1(t)$, $t \geq 1$; (b) $A^k(k)$, $k \geq 2$; (c) $A^k(t)$, $2 \leq k < t$.

The last two observations above can be easily verified by mathmatical induction based on the recursive definition.

Also it is interesting to note that the parameter k in some sense denotes the "order" of a $T^k(t)$ tree, i.e., the level of recursions that one need to go through when tracing $T^k(t)$ back to the primitive construct $T^1(1)$. It can be proved, again by induction, $T^k(t)$ has k computation nodes on its most significant column. However, the proof is omitted here since it is not crucial in the following discussion.

As an example, Figure 7(a) shows the $T^3(5)$ tree whose depth is 5, and how it is

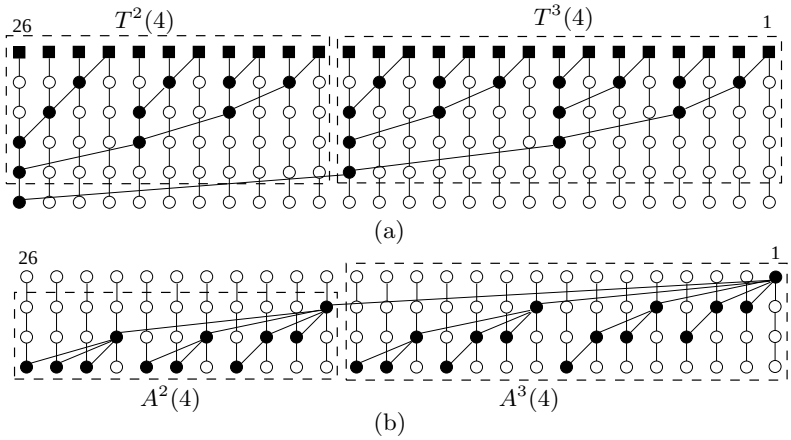

Fig. 7. Examples of the $T^k(t)$ and $A^k(t)$ trees: (a) $T^3(5)$; (b) $A^3(5)$.

Function: TtreeGen(k,t)
Input : $k \in \mathbb{Z}+$, $t \in \mathbb{Z}+$
Output : $T^k(t)$ Circuit

if $t < k$ **then**
  return *-1*;
  // $T^k(t)$ does not exist.
else if $k = 1$ **then**
  Generate $T^1(t)$ according to Figure 5(a);
  return $T^1(t)$;
else if $k = t$ **then**
  $T^{k-1}(k-1) \leftarrow$ `TtreeGen` $(k-1, k-1)$;
  Construct $T^k(k)$ according to Figure 5(b);
  return $T^k(k)$;
else
  $T^{k-1}(t-1) \leftarrow$ `TtreeGen` $(k-1, t-1)$;
  $T^k(t-1) \leftarrow$ `TtreeGen` $(k, t-1)$;
  Construct $T^k(t)$ according to Figure 5(c);
  return $T^k(t)$;
end

Algorithm 1: Generation of the $T^k(t)$ circuit.

constructed from $T^2(4)$ and $T^3(4)$, conforming to Figure 5(c). One can even trace the composition of $T^2(4)$ to $T^1(4)$ and $T^2(3)$, and similar for $T^3(4)$.

Algorithm 1, which is essentially a direct algorithmic translation of Figure 5, formally presents how a $T^k(t)$ tree is generated.

Following nearly the same recursive way of construction as the $T^k(t)$ trees, we define the $A^k(t)$ trees as shown in Figure 6. The algorithmic description of an $A^k(t)$ tree is presented in Algorithm 2. By simple induction, we can infer that for an $A^k(t)$ tree, $k+1$ is the tree depth while $t+1$ is the fan-out of the root. We also

Function: AtreeGen(k,t)
Input : $k \in \mathbb{Z}+$, $t \in \mathbb{Z}+$
Output : $A^k(t)$ Circuit

if $t < k$ **then**
  return *-1*;
  // $A^k(t)$ `does not exist.`
else if $k = 1$ **then**
  Generate $A^1(t)$ according to Figure 6(a);
  return $A^1(t)$;
else if $k = t$ **then**
  $A^{k-1}(k-1) \leftarrow$ `AtreeGen` $(k-1, k-1)$;
  Construct $A^k(k)$ according to Figure 6(b);
  return $A^k(k)$;
else
  $A^{k-1}(t-1) \leftarrow$ `AtreeGen` $(k-1, t-1)$;
  $A^k(t-1) \leftarrow$ `AtreeGen` $(k, t-1)$;
  Construct $A^k(t)$ according to Figure 6(c);
  return $A^k(t)$;
end

Algorithm 2: Generation of the $A^k(t)$ circuit.

give an example of the $A^3(5)$ tree shown in Figure 7(b). Comparing $A^3(5)$ with $T^3(5)$ helps to point out the similarity between $A^k(t)$ and $T^k(t)$. In fact, we notice that

—$A^1(t)$ and $T^1(t)$ have the same width for $t \geq 1$ (see Figure 5(a) and Figure 6(a));
—$A^k(t)$ and $T^k(t)$ shall have the same recurrence formula of the width for $2 \leq k \leq t$ (compare Figure 5(b) with Figure 6(b), and Figure 5(c) with Figure 6(c)).

Therefore, it is straightforward to conclude that $T^k(t)$ and $A^k(t)$ have the same width.

Summarizing the above discussion, we have the following theorem:

THEOREM 3.1. *Given $k \in \mathbb{Z}+$ and $t \in \mathbb{Z}+$ s.t. $1 \leq k \leq t$, the following properties regarding $T^k(t)$ and $A^k(t)$ hold:*

(1) $T^k(t)$ and $A^k(t)$ have the same width, which is

$$N(k,t) = \sum_{i=0}^{k} \binom{t}{i} \text{ for } 1 \leq k \leq t \tag{6}$$

(2) $T^k(t)$ is a binary tree of depth t and size $N(k,t) - 1$;

(3) $A^k(t)$ has a depth of $k+1$ and a size of $N(k,t)$, exactly one computation node per column.

PROOF. The derivation of Equation (6) can be found in Appendix A. (2) and (3) can be proved directly using mathematical induction which is omitted here. □

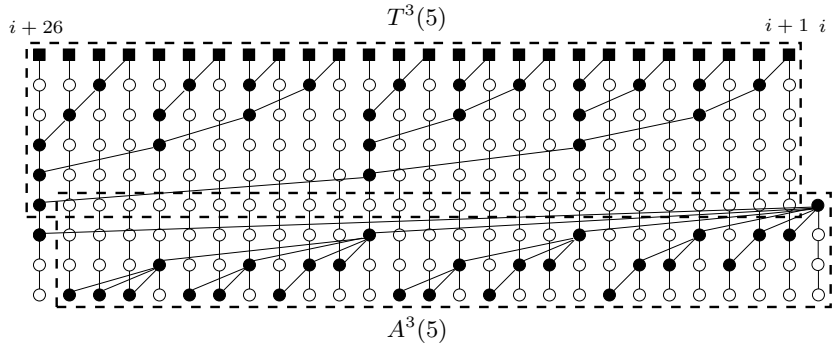

Fig. 8. Assembling $T^3(5)$ and $A^3(5)$.

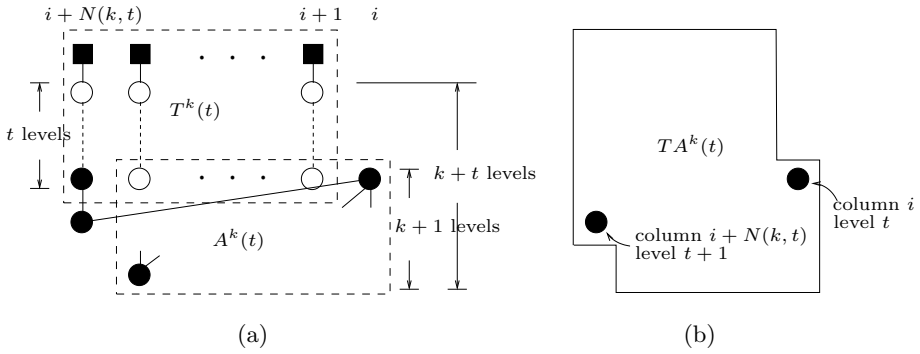

Fig. 9. (a) Assembling $T^k(t)$ and $A^k(t)$ into a prefix circuit $TA^k(t)$; (b) An abstract representation of $TA^k(t)$.

Note that when $t = k$, the width of $T^k(k)$ is $N(k,k) = \sum_{i=0}^{k} \binom{k}{i} = 2^k$, which is in accordance with our previous observation.

3.2 Construct Z(d) circuit by assembling $T^k(t)$ and $A^k(t)$

It is interesting to note that $T^3(5)$ and $A^3(5)$ can be assembled together to form a prefix circuit, as shown in Figure 8. If we consider the root of $A^3(5)$ as the final prefix output Y_i, from the figure we can tell that the assembled structure produces all the final prefix outputs Y_j for $i+1 \le j \le i+26$. In general, we can always combine a pair of a $T^k(t)$ tree and an $A^k(t)$ tree to form a prefix circuit as shown in Figure 9.

THEOREM 3.2. *Assemble $T^k(t)$ and $A^k(t)$ trees together as shown in Figure 9(a), and denote the resulting structure as $TA^k(t)$:*

(1) $TA^k(t)$ generates the final prefix outputs Y_j $(i+1 \le j \le i+N(k,t))$ if the root of $A^k(t)$ is the final prefix output Y_i;

(2) The depth of $TA^k(t)$ is $k+t$;

(3) The size of $TA^k(t)$ is $2N(k,t)$.

PROOF. We will first prove (1) by mathematical induction. The base cases of

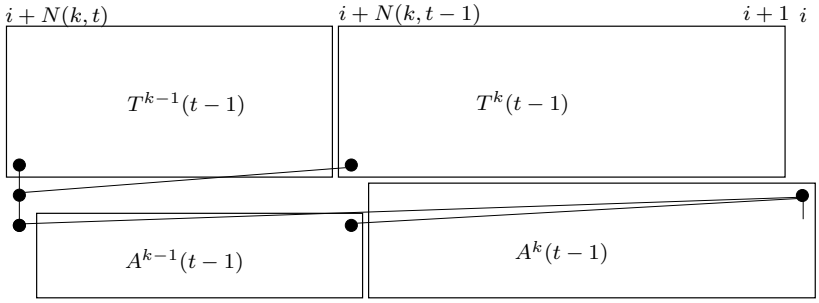

Fig. 10. A decomposed view of $TA^k(t)$.

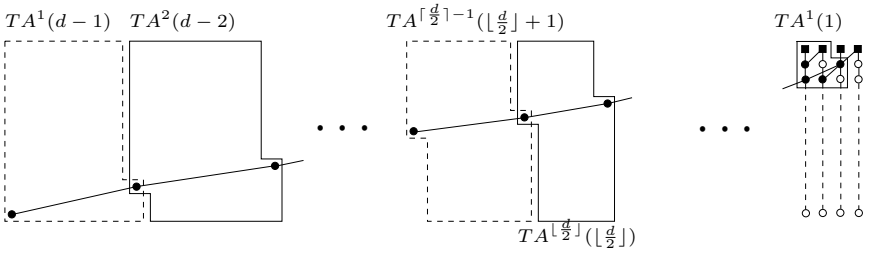

Fig. 11. A new class of zero-deficiency prefix circuits $Z(d)$.

$TA^1(1)$ and $TA^2(1)$ can be verified with ease.

Using the definitions of $T^k(t)$ and $A^k(t)$, the $TA^k(t)$ circuit defined in Figure 9(a) can be decomposed into the one shown in Figure 10. Note that if we move $A^k(t-1)$ one level toward $T^k(t-1)$ then essentially they form $TA^k(t-1)$. Therefore by inductive assumption, the final prefix outputs Y_j for $i+1 \le j \le i+N(k,t-1)$ are available.

Likewise, moving $A^{k-1}(t-1)$ two level closer to $T^{k-1}(t-1)$ would yield $TA^{k-1}(t-1)$. Since the root of $A^{k-1}(t-1)$ is the final prefix output $Y_{i+N(k,t-1)}$, the final prefix outputs Y_j for $i+N(k,t-1)+1 \le j \le i+N(k,t)$ are available too.

Second, from Figure 9(a) we can immediately tell that the depth of $TA^k(t)$ is $(k+t)$ since $T^k(t)$ and $A^k(t)$ have one level overlapping.

Finally,

$$\begin{aligned} \text{size of } TA^k(t) &= 1 + (\text{size of } T^k(t)) + (\text{size of } A^k(t)) \\ &= 1 + (N(k,t)-1) + N(k,t) \\ &= 2N(k,t) \end{aligned}$$

□

The proposed $Z(d)$ circuit is constructed by concatenating $TA^i(i)$ $(1 \le i \le \lfloor d/2 \rfloor)$ and $TA^{d-i}(i)$ $(\lfloor d/2 \rfloor < i \le d-1)$ in the increasing order of i, as illustrated in Figure 11. When concatenating two neighboring circuits $TA^{k_1}(i)$ and $TA^{k_2}(i+1)$, the most significant output of $TA^{k_1}(i)$ merges with the least significant input of $TA^{k_2}(i+1)$, allowing the former to feed the latter. Therefore, starting from $TA^1(1)$, the $Z(d)$ circuit works really like a line of dominos, generating prefix results for every

Function: ZGen(d)
Input : $d \in \mathbb{Z}+$
Output : Zero-deficiency prefix circuit $Z(d)$
if $d = 1$ **then**
  $Z(1) \leftarrow T^1(1)$;
  return $Z(1)$;
else
  for $i = 1$ **to** $\lceil \frac{d}{2} \rceil - 1$ **do**
    $T^i(d-i) \leftarrow$`TtreeGen` $(i, d-i)$; // call algorithm 1;
    $A^i(d-i) \leftarrow$`AtreeGen` $(i, d-i)$; // call algorithm 2;
    Stitch $T^i(d-i)$ and $A^i(d-i)$ into $TA^i(d-i)$ as shown in Figure 9(a);
  end
  for 1 **to** $i = \lfloor \frac{d}{2} \rfloor$ **do**
    $T^i(i) \leftarrow$`TtreeGen` (i, i); // call algorithm 1;
    $A^i(i) \leftarrow$`AtreeGen` (i, i); // call algorithm 2;
    Stitch $T^i(i)$ and $A^i(i)$ into $TA^i(i)$ as shown in Figure 9(a);
  end
  Stitch all the T, A trees together to form $Z(d)$ as shown in Figure 11;
end

Algorithm 3: Generation of the $Z(d)$ circuit.

column.

As an example, Figure 12 shows the $Z(d)|_{d=8}$ circuit, which is constructed from $TA^1(7)$, $TA^2(6)$, $TA^3(5)$, $TA^4(4)$, $TA^3(3)$, $TA^2(2)$ and $TA^1(1)$. Algorithm 3 describes the generation of the $Z(d)$ circuit, while Theorem 3.3 gives the width of the $Z(d)$ circuit.

THEOREM 3.3. *The width of the $Z(d)$ circuit, which we denote by $N_Z(d)$, is*

$$N_Z(d) = F(d+3) - 1 \text{ for } d \geq 1 \tag{7}$$

where $F(k)$ is the well-known Fibonacci series defined by the recurrence

$$F(1) = F(2) = 1, F(k) = F(k-1) + F(k-2) \quad for\ k \geq 3$$

PROOF. See appendix B □

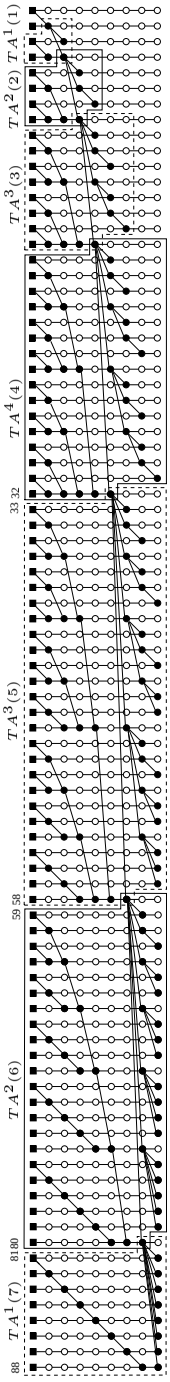

Fig. 12. An example of the $Z(d)$ circuit: $Z(d)|_{d=8}$.

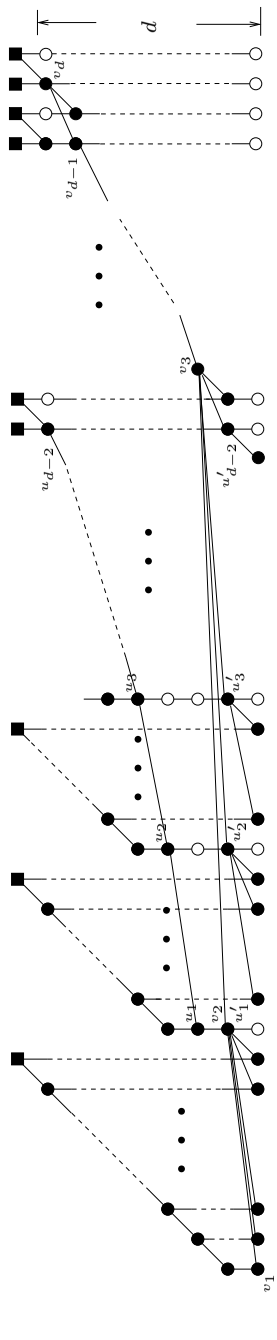

Fig. 13. Zero-deficiency prefix circuit of maximum width in action.

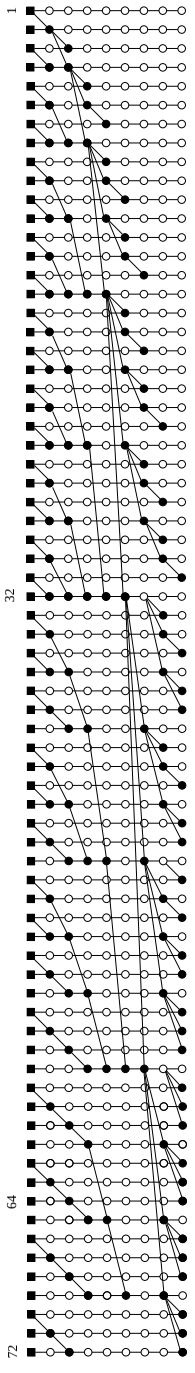

Fig. 14. A zero-deficiency prefix circuit with depth 8 and fan-out 4, derived from $Z(d)|_{d=8}$.

3.3 Optimality of the $Z(d)$ Circuit

In order to prove the optimality of the $Z(d)$ circuit, we shall first show that the $Z(d)$ circuit is indeed of zero-deficiency, which is addressed in Theorem 3.4, and prove that it does have the minimum depth among all possible zero-deficiency circuits, which is addressed in Theorem 3.5. Actually, the intuition behind constructing the $Z(d)$ circuit is that we can try to find the zero-deficiency prefix circuit of maximum width for a given depth, instead of finding the zero-deficiency prefix circuit of minimum depth for a given width. Of course, the two points of view are actually equivalent.

THEOREM 3.4. *The parallel prefix circuit $Z(d)$ shown in Figure 11 is of zero-deficiency.*

PROOF. The depth of $Z(d)$ is automatically d by the way we constructed it. The size of $Z(d)$ is the sum of the sizes of all the constituent $TA^k(t)$ circuits. Note that every two neighboring $TA^k(t)$ circuits have one node in common, and there are in total $d-2$ common nodes. Therefore,

$$\begin{aligned} s_{Z(d)} &= \sum (\text{size of } AT^k(t) \text{ circuits}) - (d-2) \\ &= 2(N_Z(d)-2) - (d-2) \\ &= 2N_Z(d) - d - 2 \end{aligned}$$

That is:

$$s_{Z(d)} + d = 2N_Z(d) - 2$$

Therefore, $Z(d)$ is of zero-deficiency. □

THEOREM 3.5. *$Z(d)$ has the maximum width for a given depth d among all zero-deficiency prefix circuits.*

PROOF. We will only sketch the arguments. We would like to show that given a depth d, the widest zero-deficiency circuit that we can build is actually $Z(d)$.

Suppose $G(n)$ is a zero-deficiency circuit of depth d. According to Corollary 2.3, the ridge of $G(n)$ has d nodes on it, one node per level. Starting from the last final prefix output Y_n, let us number the nodes on the ridge as v_1,v_2,$\cdots$,v_d in order, as shown in Figure 13, and correspondingly denote the columns where they reside in as col_{v_1}, col_{v_2}, $\cdots$, col_{v_d}. Node v_i is at level $d-i+1$.

(1) For columns between col_{v_1} and col_{v_2}, the only way to get the prefix results is to take node v_2 as the input and produce the results at level d. Accordingly, the only possible structure above the $v_1 - -v_2$ connection is a serial prefix circuit, and its maximum possible width is d.

(2) Now consider the columns between col_{v_2} and col_{v_3}. The structure above the $v_2 - -v_3$ connection is an alphabetical binary tree, and its root is the node directly above v_2. For this tree to have a span as wide as possible, the path from its root to its least significant input must increase exactly one level each time. The total number of nodes on this path is $d-2$, and every node is a partial prefix result. Let us label these nodes as u_1, u_2,$\cdots$, u_{d-2}, and their corresponding columns as col_{u_1}, col_{u_2},$\cdots$,$col_{u_{d-2}}$. $Z(d)$'s prefix results at columns col_{u_i} $(i = 1 \cdots d-2)$ can be generated at level $d-1$ by directly taking v_3 as the input.

Table I. Widths of $LS(n)$, $LYD(n)$ and $Z(d)$ circuits.

depth	W_{LS}	W_{LYD}	W_Z	depth	W_{LS}	W_{LYD}	W_Z
3	7	7	7	13	260	308	986
4	11	12	12	14	383	446	1596
5	16	20	20	15	517	576	2583
6	23	33	33	16	575	843	4180
7	33	54	54	17	1030	1101	6764
8	47	77	88	18	1535	1625	10945
9	66	95	143	19	2055	2139	17710
10	95	135	232	20	3071	3176	28656
11	131	169	376	21	4104	4202	46367
12	191	242	609	22	6143	6264	75024

(3) Again, label these output nodes at level $d-1$ as u'_1, u'_2,$\cdots$, u'_{d-2} (note u'_1 is just v_2). The outputs of the columns between u'_1 and u'_2 have no other choice but to take u'_2 as the input directly. Accordingly, the structure above the $u_1 - -u_2$ connection must be a serial prefix circuit, and its maximum width is $d-2$. Likewise, the structure above the $u_2 - -u_3$ connection is also a prefix circuit whose maximum width is $d-3$, and so on. Therefore, the binary tree above the $v_2 - -v_3$ connection is identical to the $T^2(d-2)$ tree.

We can continue the above analysis to the lower columns in a similar way. It turns out that in order to make the whole zero-deficiency circuit as wide as possible, the binary tree above the $v_3 - -v_4$ connection must be the $T^3(d-3)$ tree, and the structure under the $v_3 - -v_4$ connection is just an $A^3(d-3)$ tree, and so on. Thus, the resulting circuit is exactly $Z(d)$ as we defined in Figure 11. □

Table I shows the widths of $Z(d)$ circuits for $3 \le d \le 22$, with the results of Lin's design [Lin and Shih 1999] and the $LYD(n)$ circuit [Lakshmivarahan et al. 1987] listed for comparison. Each number is read as the maximum width up to which a zero-deficiency prefix circuit of that type under the given depth can be constructed. Clearly, our design dominates the other two, especially when the depth is large.

4. GENERALIZED $Z(D)$ CIRCUIT

It is now clear that the curve of function $d = f(n)$ in Figure 2 is just the inverse function of $N_{Z(d)} = F(d+3) - 1$ given in Theorem 3.3. Formally, we have

THEOREM 4.1. *The minimum depth of any n-input zero-deficiency prefix circuit is given by*

$$d_{min}(n) = \min\{t : F(t) \ge n+1\} - 3 \tag{8}$$

Figure 15 sketches the shape of $d_{min}(n)$, which is a piecewise constant function. Note that the $Z(d)$ circuits described in Algorithm 3 only represent the extreme cases of the zero-deficiency circuits, namely, the rightmost points of the step segments in Figure 15. It is of practical interest to construct zero-deficiency circuits for every width-depth pair (n, d) such that $d \ge d_{min}(n)$. Fortunately, this is not difficult to do based on the proposed $Z(d)$ circuit, and can be tackled in two steps.

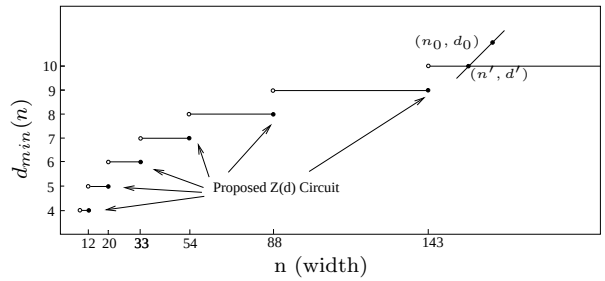

Fig. 15. $d_{min}(n)$: minimum depth of zero-deficiency prefix circuits as a function of width n.

Function: ZXGen(n,d)
Input : $n \in \mathbb{Z}+$, $d \in \mathbb{Z}+$
Output : A zero-deficiency prefix circuit ZX(n,d) of width n and depth d
if $d < d_{min}(n)$ **then**
  return *-1*;
else
  Find t_0 s.t. $d - t_0 = d_{min}(n - t_0)$;
  $Z(d - t_0) \leftarrow$`ZGen` $(d - d_0)$; // Call Algorithm 3;
  $ZX(n - t_0, d - t_0) \leftarrow$truncating the t_0 most significant columns of $Z(d - t_0)$;
  $ZX(n, d) \leftarrow$Concatenate $ZX(n - t_0, d - t_0)$ with a t_0-input serial prefix circuit as shown in Figure 16;
  return *ZX(n,d)*;
end

Algorithm 4: Generation of zero-deficiency circuit for arbitrary (n, d) pair.

First, we need to consider all the (n_0, d_0) pairs residing on the curve of $d_{min}(n)$. For these points, we can simply construct $Z(d)|_{d=d_0}$ first, whose width is $F(d_0 + 3) - 1$, and discard the most significant $F(d_0 + 3) - 1 - n_0$ columns. For example, a 64-input prefix circuit of depth 8 can be obtained by discarding the leading 24 columns of $Z(d)|_{d=8}$. Care must be taken to make small adjustments on the new most significant columns after discarding.

For (n_0, d_0) pairs such that $d_0 > d_{min}(n_0)$, we first draw a 45 degree line though (n_0, d_0), intersecting $d_{min}(n)$ at (n', d'), as shown in Figure 15. We can then construct zero-deficiency circuit for (n', d'), and concatenate it with a $(d_0 - d')$-input serial prefix circuit.

The above idea translates into Algorithm 4, which generates zero-deficiency prefix circuits for arbitrary (n, d) pairs.

5. COMPARISONS AND DISCUSSIONS

5.1 A comparison to Zimmermann's algorithm

In [1996], Zimmermann proposed a deterministic algorithm to generate parallel prefix circuit of small size under a given depth constraint. His algorithm consists of two steps: compressions and expansion. The compression step starts with a serial prefix circuit and compact it as much as possible, effectively reducing the depth at

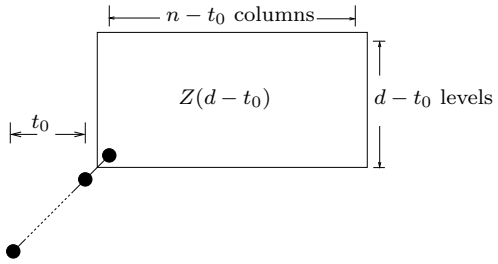

Fig. 16. Zero-deficiency prefix circuit for arbitrary (n, d) pair; t_0 is a positive integer such that $d - t_0 = d_{min}(n - t_0)$.

the cost of increasing size. The expansion step tries to reduce the size by bouncing back the depth subject to the depth constraint. His algorithm in many cases can produce zero-deficiency prefix circuit. However, since his algorithm is greedy in nature, there is no guarantee on the optimality. In contrast, the proposed $Z(d)$ circuit and its generalization has provable optimality for $d \geq d_{min}(n)$, and shows its advantage when the depth constraint is really tight. For example, given width 54 and depth 7, our design has 99 nodes, while Zimmermann's algorithm gives a design of 104 nodes [Zimmermann].

Another merit of the proposed structure is that, when the depth constraint is loose, Algorithm 4 tends to compact all the nodes to lower levels and columns. Such a scheme can achieve low power in the context of prefix adder design, because cells of higher logic levels usually have higher activity rates which are proportional to dynamic power consumption. Zimmermann's algorithm, however, tends to bring nodes to higher levels in the expansion step. A detailed analysis on power characteristic of the proposed structure is planned for future work.

On the other hand, Zimmermann's algorithm can handle cases of integer, non-uniform input arrival times (AT), while the proposed approach only considers the case of uniform AT. Hence it is interesting to extend our idea to produce *provably* good prefix circuits under non-uniform AT. A conceivable way to attack this problem is to partition the whole circuit into consecutive sub-blocks, each with an uniform AT profile, and solve the sub-blocks individually and sequentially. In this case, each sub-block may receive a delayed first input, which is generated by the previous sub-block.

5.2 Zero-deficiency prefix circuits of limited fan-out

Another possible extension problem is to generalize the $Z(d)$ circuit and consider fan-out constraints. Lin et. al. has done a lot of work in this respect, that is, constructing zero-deficiency[1] prefix circuits of limited fan-out, especially 2 and 4 [Lin 1999][Lin et al. 2003][Lin and Chen 2003][Lin and Hsiao 2004]. Their approaches are largely constructive and cannot be generalized for arbitrary fan-out constraints. However, we may have a different way of constructing fan-out limited prefix circuits by directly pruning the out branches of large fan-out nodes in our proposed

[1]Lin et. al. had used depth-size optimal to denote the concept of zero-deficiency in their papers. However, we explicitly distinguish depth-size optimal and zero-deficiency, as stated in Section 1.

$Z(d)$ circuit. A preliminary study shows that by doing this way we can generate a prefix circuit of up to 72-bit for depth constraint 8 and fan-out limit 4, as shown in Figure 14. By contrast, under the same constraints the maximum widths that can be achieved by H4 [Lin et al. 2003], Z4 [Lin and Chen 2003] and WE4 [Lin and Hsiao 2004] are 64, 67 and 57 respectively.

6. CONCLUSIONS

In this paper, we have proposed a new class of zero-deficiency prefix circuits $Z(d)$ which has the minimum possible depth for a given width. We described the construction of $Z(d)$ and proved its optimality in detail. We also extend $Z(d)$ circuits to consider general width and depth specifications. Thus for $d \geq d_{min}(n)$ (as established in Theorem 4.1), the optimal depth-size tradeoff problem for prefix circuits is completely resolved.

APPENDIX

A. DERIVATION OF EQUATION (6)

PROOF. According to the definition of $T^k(t)$, $N(k,t)$ obeys the following recurrence formula:

$$N(k,t) = N(k,t-1) + N(k-1,t-1) \quad \text{for } 2 \leq k < t \tag{9}$$

and boundary conditions:

$$N(1,t) = t \quad \text{for } t \geq 1 \tag{10}$$

$$N(k,k) = 2^k \quad \text{for } k \geq 2 \tag{11}$$

Although $N(k,t)$ is defined over $1 \leq k \leq t$ only, mathematically it is valid to extend $N(k,t)$ to the entire first quadrant of $t \geq 0, k \geq 0$. Interestingly if we set the new boundary conditions to be

$$N(0,t) = N(k,0) = 1 \quad \text{for } t \geq 0, k \geq 0 \tag{12}$$

Then the extended $N(k,t)$ sequence still satisfies the recurrence equation (9) for $t \geq 1, k \geq 1$, as well as the original boundary conditions (10) and (11).

We can then construct a generating function using the extended $N(k,t)$ as the corresponding coefficients:

$$F(x,y) = \sum_{\substack{k \geq 0 \\ t \geq 0}} N(k,t) x^k y^t \tag{13}$$

We can further derive that

$$\begin{aligned} F(x,y) &= \sum_{k\geq 0} x^k + \sum_{t\geq 1} y^t + \sum_{\substack{k\geq 1\\ t\geq 1}} N(k,t)x^k y^t \\ &= \sum_{k\geq 0} x^k + \sum_{t\geq 1} y^t + \sum_{\substack{k\geq 1\\ t\geq 1}} (N(k,t-1)+N(k-1,t-1))x^k y^t \\ &= \sum_{k\geq 0} x^k + y\sum_{t\geq 0} y^t + y\sum_{\substack{k\geq 1\\ t\geq 0}} N(k,t)x^k y^t + xy\sum_{\substack{k\geq 0\\ t\geq 0}} N(k,t)x^k y^t \\ &= \sum_{k\geq 0} x^k + yF(x,y) + xyF(x,y) \end{aligned} \tag{14}$$

Thus

$$F(x,y) = \frac{1}{1-y(1+x)} \cdot \sum_{k\geq 0} x^k = \sum_{k\geq 0} (y(1+x))^k \cdot \sum_{k\geq 0} x^k \tag{15}$$

Recall that $N(k,t)$ is the coefficient of term $x^k y^t$ in $F(x,y)$. The only term containing y^t in the right-hand side of (15) is $y^t(1+x)^t$, and

$$(1+x)^t = \sum_{i=0}^{t} \binom{t}{i} x^i \tag{16}$$

Clearly, only the first $k+1$ terms of the expansion of $(1+x)^t$ will generate x^k when multiplied by proper terms in $\sum_{k\geq 0} x^k$. Therefore

$$N(k,t) = \sum_{i=0}^{k} \binom{t}{i} \quad \text{for } 1 \leq k < t \tag{17}$$

□

B. PROOF OF THEOREM 3.3

PROOF. From Figure 11 we see that the width of $Z(d)$ is just the sum of the widths of all its constituent $TA^k(t)$ circuits plus 2.

$$\begin{aligned} N_Z(d) &= 2 + \sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} N(i,i) + \sum_{i=\lfloor \frac{d}{2} \rfloor + 1}^{d-1} N(d-i,i) \\ &= 2^{\lfloor \frac{d}{2} \rfloor + 1} + \sum_{i=1}^{\lceil \frac{d}{2} \rceil - 1} N(i,d-i) \end{aligned} \tag{18}$$

What remains is to derive a closed form for the right-hand side of the above equation. In writing $N_Z(d)$ we will distinguish the cases of d even and d odd.

$$N_Z(2d) = 2^{d+1} + \sum_{i=1}^{d-1}\sum_{j=0}^{i}\binom{2d-i}{j} \tag{19}$$

$$N_Z(2d+1) = 2^{d+1} + \sum_{i=1}^{d}\sum_{j=0}^{i}\binom{2d+1-i}{j} \tag{20}$$

Taking the difference of $N_Z(2d+1)$ and $N_Z(2d)$, and utilizing the well-known binomial equation $\binom{n}{k} - \binom{n-1}{k} = \binom{n-1}{k-1}$, we have

$$\begin{aligned}
N_Z(2d+1) - N_Z(2d) &= \sum_{i=1}^{d}\sum_{j=0}^{i}\binom{2d+1-i}{j} - \sum_{i=1}^{d-1}\sum_{j=0}^{i}\binom{2d-i}{j} \\
&= \sum_{j=0}^{d}\binom{d+1}{j} + \sum_{i=1}^{d-1}\sum_{j=0}^{i}\left(\binom{2d+1-i}{j} - \binom{2d-i}{j}\right) \\
&= (2^{d+1}-1) + \sum_{i=1}^{d-1}\sum_{j=1}^{i}\binom{2d-i}{j-1} \\
&= 2^{d+1} + \sum_{i=1}^{d-1}\sum_{j=0}^{i}\binom{2d-i}{j} - 1 - \sum_{i=1}^{d-1}\binom{2d-i}{i} \\
&= N_Z(2d) - \sum_{i=0}^{d}\binom{2d-i}{i} + 1
\end{aligned} \tag{21}$$

Therefore

$$2N_Z(2d) - N_Z(2d+1) = \sum_{i=0}^{d}\binom{2d-i}{i} - 1 \tag{22}$$

Similarly, we have

$$2N_Z(2d+1) - N_Z(2d+2) = \sum_{i=0}^{d}\binom{2d+1-i}{i} - 1 \tag{23}$$

It is easy to show by induction that

$$\sum_{i=0}^{d}\binom{2d-i}{i} = F(2d+1) \tag{24}$$

$$\sum_{i=0}^{d}\binom{2d+1-i}{i} = F(2d+2) \tag{25}$$

where $F(r)$ denotes the r^{th} Fibonacci number. Now let us substitute Equations (24) and (25) into Equations (22) and (23) respectively, and then unify these two equations, we obtain

$$N_Z(d+1) = 2N_Z(d) - F(d+1) + 1 \tag{26}$$

Again, it can be shown by induction that

$$N_Z(d) = F(d+3) - 1 \text{ for } d \geq 1 \tag{27}$$

□

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

Bilgory, A. and Gajski, D. D. 1986. A heuristic for suffix solutions. *IEEE Trans. Comput. 35,* 1 (Jan.), 34–42.

Brent, R. P. and Kung, H. T. 1982. A regular layout for parallel adders. *IEEE Trans. Comput. 31,* 3 (Mar.), 260–264.

Fich, F. E. 1983. New bounds for parallel prefix circuits. In *Proc. of the 15th Annual ACM Symposium on Theorey of Computing (STOC '83)*. ACM Press, New York, NY, USA, 100–109.

Ladner, R. E. and Fischer, M. J. 1980. Parallel prefix computation. *Journal of the Association for Computing Machinery 27,* 4 (Oct.), 831–838.

Lakshmivarahan, S. and Dhall, S. K. 1994. *Parallel Computing: Using the Prefix Problem*. Oxford University Press, New York.

Lakshmivarahan, S., Yang, C. M., and Dhall, S. K. 1987. On a new class of optimal parallel prefix circuits with (size+depth)=$2n-2$ and $\lceil \log n \rceil \leq depth \leq (2\lceil \log n \rceil - 3)$. In *Proc. of the 1987 Int. Conf. on Parallel Processing*. 58–65.

Lin, Y. C. 1999. Optimal parallel prefix circuits with fan-out 2 and corresponding parallel algorithms. *Neural, Parallel and Scientific Computations 7,* 1 (Mar.), 33–42.

Lin, Y. C. and Chen, J. N. 2003. Z4: A new depth-size optimal parallel prefix circuit with small depth. *Neural, Parallel and Scientific Computations 11,* 3 (Sept.), 221–235.

Lin, Y. C. and Hsiao, J. W. 2004. A new approach to constructing optimal parallel prefix circuits with small depth. *Journal of Parallel and Distributed Computing 64,* 1 (Jan.), 97–107.

Lin, Y. C., Hsu, Y. H., and Liu, C. K. 2003. Constructing h4, a fast depth-size optimal parallel prefix circuit. *Journal of Supercomuting 24,* 3 (Mar.), 279–304.

Lin, Y. C. and Shih, C. C. 1999. A new class of depth-size optimal parallel prefix circuits. *Journal of Supercomputing 14,* 1 (July), 39–52.

Sklansky, J. 1960. Conditional sum addition logic. *IRE Trans. Electron. Comput EC-9,* 6 (June), 226–231.

Snir, M. 1986. Depth-size trade-offs for parallel prefix computation. *Journal of Algorithm 7,* 2 (June), 185–201.

Zimmermann, R. Java applet for adder synthesis. In *http://www.iis.ee.ethz.ch/ zimmi/ applets/ prefix.html*.

Zimmermann, R. 1996. Non-heuristic optimization and synthesis of parallel-prefix adders. In *Int. Workshop on Logic and Architecture Synthesis*. 123–132.

Received August 2004; Revised February 2005 and August 2005; accepted January 1996.