CHALMERS

A new approach to the design of optimal parallel prefix circuits

MARY SHEERAN
IAN PARBERRY

Technical Report No. 2006:1

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2006

A new approach to the design of optimal parallel prefix circuits

Mary Sheeran
Chalmers University of Technology
ms@cs.chalmers.se

Ian Parberry
University of North Texas
ian@unt.edu

February 28, 2006

Abstract

Parallel prefix is one of the fundamental algorithms in computer science. Parallel prefix networks are used to compute carries in fast addition circuits, and have a number of other applications, including the computation of linear recurrences and loop parallelization. A new construction, called *Slices*, for fan-out-constrained depth size optimal (DSO) parallel prefix circuits is presented. The construction encompasses the largest possible number of inputs for given depth and fan-out. The construction improves on previous approaches that produce DSO networks with constrained fan-out by encompassing more inputs for a given depth. Even when compared with parallel prefix circuits with unbounded fan-out, the construction provides a new family of circuits that are both small and reasonably shallow. We present the construction, which is composed of recursively defined blocks, and derive a recurrence for the maximum number of inputs that can be processed for a given fan-out and depth. We also show how a DSO network built according to our construction can be cropped, to produce a new DSO network with the same depth and fan-out, but fewer inputs. Thus, we can produce a DSO network for given depth, fan-out and number of inputs, provided such a network exists. We believe that we are the first to be able to do this. The resulting networks are compared to others with both bounded and unbounded fan-out.

1 Introduction

The *prefix problem* is to compute all the products $x_1 \circ x_2 \circ \ldots x_k$ for $1 \leq k \leq n$, for an associative operator $\circ$. The most obvious solution is to connect a series of $n-1$ operators in a linear array, as shown schematically in Figure 1(a). Input nodes are on the top of the circuit, with the least significant input (x_1) being on the left. Data flows from top to bottom, and we also count the stages or levels of the circuit in this direction, starting with level zero on the top. An operation node, represented by a small circle, performs the $\circ$ operation on its two inputs. One of the inputs comes along the diagonal line above and to the left of the node, and the other along the vertical line feeding the node from the top. A node always produces an output to the bottom along the vertical line. It may also produce an output along a diagonal line below and to the right of the node. Here, at level zero, there is a diagonal line leaving a vertical wire in the absence of a node. This is a fork. The diagram shows a ripple-carry structure with 8 inputs. We say that the circuit has *width* 8. The circuit shown contains 7 nodes, and so is said to be of *size* 7. Its lowest level in the picture is level 7, so the circuit has *depth* 7. The depth of a circuit is the maximum number of operators on a path from input to output. In real circuits, this is related to delay, where the time unit is the the time taken to perform one operation. So one can think of the levels in the diagrams as time steps.

The fan-out of a node is its out-degree, and of a circuit is the maximum of the fan-outs of its nodes. In this example, all but the rightmost node have fan-out 2, so the whole circuit is said to have fan-out 2. We will typically use n for number of inputs, s for size, d for depth and f for

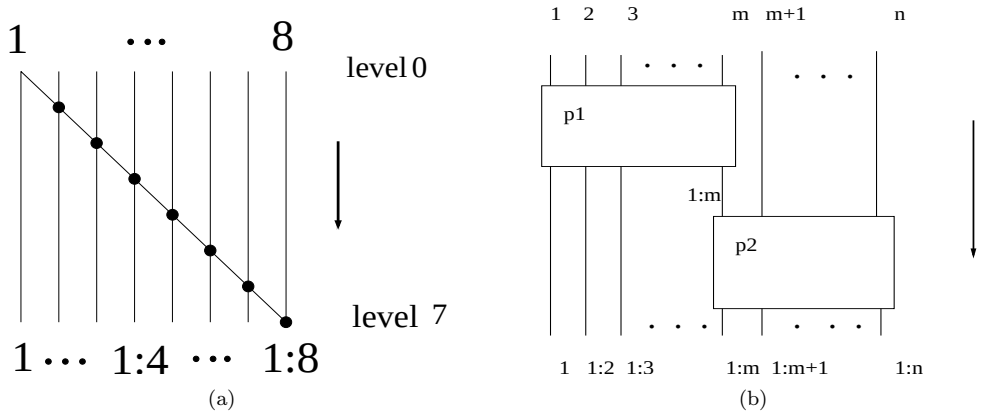

Figure 1: (a) The serial prefix structure (b) Construction of a parallel prefix circuit by composition of two smaller such circuits, p1 and p2. The arrows indicate direction of data flow. The least significant input is at top left, and the most significant output at bottom right.

fan-out. To ease presentation, we write $1:k$ for $x_1 \circ x_2 \circ \ldots \circ x_k$. Note that our rules of construction mean that, even in sub-circuits that are not themselves parallel prefix networks, output i always depends on input i, and possibly also on inputs with index lower than i. We will assume that this is the case in what follows.

For two inputs, there is only one reasonable way to construct a prefix circuit, using one copy of the operator. Prefix circuits can also be formed by composing two smaller such circuits, as shown in Figure 1(b). Repeated application of this pattern (and the base case) produces the serial prefix circuit that we have already seen.

Examining the serial prefix structure in Figure 1(a), we see that at each non-zero level only one of the vertical lines contains an operator. *Parallel* prefix circuits can have more than one operator at a given level, so that there is some parallelism in the resulting computation. Parallel prefix structures have been much studied because they shed light on the theory of parallelism, and on the complexity of computation by networks [21].

Parallel prefix networks are built, in practice, to perform the computation of carries in addition circuits [30, 10]. The study of how to compute carries in binary adders has been going on since the late 1950s, and the old papers are fascinating. One of them proposes what is now called the Manchester carry chain [11] – an approach to avoiding long carry propagations that is still used. The original paper describes the construction of an 18 stage adder, with the stages on individual plug-in boards. The authors state that this "results in a transmission delay of approximately 20 millimicrosec. It is thought that careful design of the boards to minimize the 5 ft length of wire in the 'carry' path will reduce this delay probably by a factor of 2.". We still have problems with wires in circuit design, but they are on a different scale!

Logarithmic depth adders were known (and built) in the early 1960s [26, 22, 17, 2] but the clean separation of the parallel prefix carry computation from the remainder of the adder came later [4, 13].

Parallel prefix circuits also turn up in other guises, such as in the implementation of priority encoders, which are important components of modern microprocessors. Parallel prefix, which is also known as *scan*, plays an important rôle in parallel programming [3]. There is an intimate connection between loop parallelization and parallel prefix [18, 25]. Hinze's presentation of the algebra of scan is an excellent introduction to the topic of parallel prefix networks [9]. In this paper, we will concentrate on the construction of parallel prefix networks at an abstract level, and will not delve into the details of how such networks are used in applications.

The minimal depth for a parallel prefix network with n inputs is $\lceil \log_2 n \rceil$. This bound is achieved by a simple divide and conquer construction. Sklansky proposed a divide and conquer

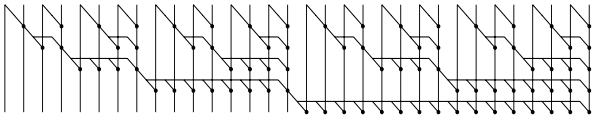

Figure 2: The Sklansky construction for 32 inputs. It recursively computes the parallel prefix for each half of the inputs and then combines the last output of the lower (left) half with each of the outputs of the upper (right) half.

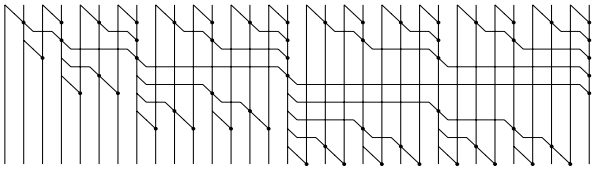

Figure 3: The Brent Kung construction with fan-out 2, for 32 inputs.

approach in 1960 in the context of conditional-sum addition [22], and so the recursive construction shown in Figure 2 for 32 inputs is usually named after him. The shallow depth is achieved at the expense of a high fan-out of $\lceil n/2 \rceil + 1$ for n inputs. For $n = 2^t$, the number of operators is $t2^{t-1} = (n/2)\log_2 n$. In Figure 2, note how at each level exactly half of the wires have operators on them.

The basic Brent Kung construction [4] is about twice as deep as Sklansky, having depth $2\log_2 n - 1$ for $n = 2^t$ inputs. However, it has fan-out of only 2, as shown in Figure 3. The Brent Kung construction also has significantly fewer operators. For $n = 2^t$ inputs, the number of operators is $\sum_i^t (2^i - 1) = 2^{t+1} - t - 2 = 2n - \log_2 n - 2$. Brent and Kung elucidated the use of parallel prefix networks in adders. The recursive pattern itself was known earlier [19, 21].

Ladner and Fischer's ingenious construction [13] allows a kind of sliding scale between a minimum depth circuit that contains fewer operators than Sklansky and one that resembles Brent Kung, being of the same size but with slightly lower depth because the fan-out is not constrained to be exactly 2. Often, in the literature, the Sklansky construction is attributed to Ladner and Fischer, and the subtlety of their trading of size against depth is thus ignored. Ladner and Fischer were fully aware of "the immediate recursive construction" and wished to improve upon it. The minimum depth Ladner Fischer network, P_0, is shown in Figure 4. For $n = 2^t$ inputs, it has $4n - F(5 + \log_2 n) + 1$ operators, where $F(m)$ is the m^{th} Fibonacci number. Thus, the size is bounded by $4n$ (but remember that size means number of operators, rather than being a more direct measure of area).

For 32 inputs, P_0 has 74 operators, compared to 80 for Sklansky and 57 for Brent Kung. For 64 inputs, P_0 has 168 operators, while Sklansky has 192. Both have depth 6 and fan-out 33. On the other hand, Brent Kung has 120 operators, depth 11 and fan-out 2. The other well-known minimum depth construction is due to Kogge and Stone [12]. It has fan-out of only 2 but requires a large number of operators (for example 321 operators for 64 inputs).

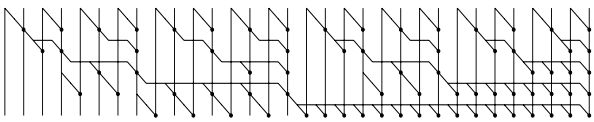

Figure 4: The minimum depth Ladner Fischer network, P_0, for 32 inputs.

1.1 The notion of depth size optimality

It is already apparent that the design space of parallel prefix circuits is large. One approach is to focus attention on the *depth size optimal* prefix networks. Snir [23] proved that for an n input parallel prefix network of size s and depth d, $d + s \geq 2n - 2$, leading to the following definition:

Definition 1.1 (Depth Size Optimality)**.** *An n input parallel prefix network of size s and depth d is depth size optimal (DSO) if $d + s = 2n - 2$.*

DSO circuits are interesting because they allow us both to build networks of small size (roughly $2n$ operators) and to trade off size against depth. Of the networks that we have seen so far, only the serial prefix is depth size optimal. The Brent Kung construction comes close, however. For $n = 2^t$ inputs, we find that $s = 2n - \log_2 n - 2$ and $d = 2\log_2 n - 1$, so that their sum is $2n + \log_2 n - 3$. This misses the $2n - 2$ bound by only $\log_2 n - 1$.

The amount by which a network misses the depth size lower bound is called its *deficiency*, and DSO circuits are therefore also called *zero-deficiency* [23]. A parallel prefix network is said to be *optimal* if it has minimum size for the given depth, number of inputs and fan-out restriction [6]. All of the circuits produced by our construction are optimal in this sense, because they are DSO. Note, however, that there are optimal parallel prefix networks that are not DSO; if the depth constraint is tight enough, then DSO circuits no longer exist, and indeed the size of the networks increases rapidly as depth is reduced further. In this report, we restrict ourselves to the study of DSO networks.

Zhu et al have recently presented an approach to the design of zero-deficiency parallel prefix circuits that closely resembles ours [27, 28]. Their construction produces zero-deficiency networks of the smallest possible depth. It does not have strictly constrained fan-out and this is the main difference in our approach. Ours is an independently derived, more general construction. We are particularly interested in circuits with small fixed fan-out that does not increase with n or with the depth, and one could view Zhu's construction as an instance of ours in which fan-out is set to be the maximum allowed by the construction, that is one greater than the depth. It is interesting to read the papers by Zhu et al, because their way of thinking about and analysing the networks is completely different from ours; yet they come to very similar conclusions. We will return to a comparison of our results with those of Zhu et al in section 5.2.2[1].

We have been inspired by a series of articles by Lin and his coworkers on the construction of DSO networks with fan-out 4 [15, 14, 16]. Later, in section 5.2.3, we will show how our construction improves upon the previously best known such networks.

In section 2, we show how to construct DSO circuits with fan-out 2. For a given depth, the construction gives a widest possible DSO circuit of that depth. Section 3 extends the previous construction to fan-out greater than 2; the maximum width that can be achieved for a given depth and fan-out is calculated. Section 4 shows how to take a maximum width DSO circuit and crop it to a given smaller width, while maintaining the depth, maximum fan-out and DSO property. Section 5 analyses the results, comparing to other approaches, and the final two sections discuss future work and conclusions.

2 The Slices construction for fan-out 2

In this section, we consider the design of DSO circuits with fan-out 2. We call the resulting construction Slices, because it is composed of similar sub-circuits, each of which can be thought of as a slice of the circuit. Each of these slices is itself a parallel prefix circuit, and we first consider how to construct these sub-blocks.

[1] Note that our use of the term depth size optimal is different from that in Zhu et al (but follows references [15, 14, 16]). Zhu et al take depth size optimal to mean having minimum size for a given depth constraint and number of inputs. What Zhu et al call zero-deficiency is the same as our definition of depth size optimality.

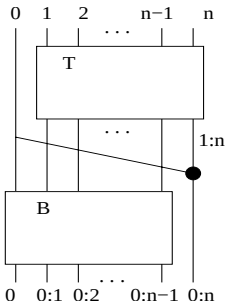

Figure 5: The general form of a WSO1 parallel prefix circuit for fan-out 2: a slice.

2.1 The notion of the waist of a parallel prefix network

The first (or leftmost) input to a parallel prefix network is always forked at least once, since the second and all subsequent outputs depend upon it. We call the level at which the first such fork occurs the *firstFork* of the network. The level at which the last output is computed is *lastOp*. The difference between *lastOp* and *firstFork* is the *waist* of the circuit. Lin et al [15] proved that for an n input parallel prefix network of waist w and size s, $w + s \geq 2n - 2$. They call a parallel prefix network that meets the lower bound *size optimal*. However, we rename the property to *waist size optimal* in order to emphasise the rôle of our notion of waist. We will call a waist-size optimal circuit WSO, and those with waist one WSO1.

Definition 2.1 (Waist Size Optimality)**.** *An n input parallel prefix network of waist w and size s is waist size optimal (WSO) if $w + s = 2n - 2$.*

2.2 Composing waist size optimal circuits

Waist size optimality is preserved by composition. Suppose we have a parallel prefix circuit p constructed from two WSO circuits p_1, p_2 as shown in Figure 1(b). Then, p_1 has m inputs, p_2 has $n - m + 1$ inputs, and p has n inputs. If p_1 has size s_1 and waist w_1, and p_2 has size s_2 and waist w_2, then p has size $s = s_1 + s_2$ and waist $w = w_1 + w_2$. Since both p_1, p_2 are WSO,

$$\begin{aligned} w_1 + s_1 &= 2m - 2 \\ w_2 + s_2 &= 2(n - m + 1) - 2 \end{aligned}$$

Therefore,

$$\begin{aligned} w + s &= (w_1 + w_2) + (s_1 + s_2) \\ &= (w_1 + s_1) + (w_2 + s_2) \\ &= (2m - 2) + (2(n - m + 1) - 2) \\ &= 2n - 2, \end{aligned}$$

which shows that p is also WSO.

2.3 Building a slice with fan-out 2

WSO1 networks are useful in building larger parallel prefix circuits, so we study their construction. A slice of fan-out 2 is a WSO1 parallel prefix network, and so must have the form shown in Figure 5. The rightmost output of the upper sub-circuit (labelled T for top) depends on each of the n inputs to T. T must, therefore, contain at least $n - 1$ operators. The situation is reversed for the lower sub-circuit, B (for bottom). Each of its outputs depends on its leftmost input, which

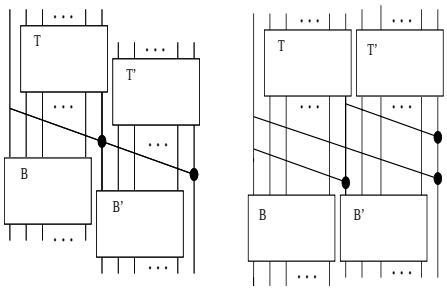

Figure 6: Composing two slices and then replacing the two operators in the middle by an equivalent three operator sub-circuit, giving a new slice, with all slices being of fan-out 2.

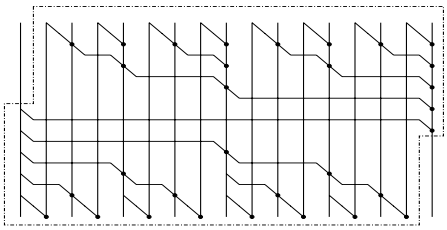

Figure 7: A fan-out 2 slice with top and bottom trees of depth 4. The dotted line shows the block symbol used to represent slices in later constructions.

is independent from all of its other inputs, in that it has not been processed by the upper tree T. Thus, this lower tree must also contain at least $n-1$ operators. Because the circuit is WSO1, its size must be $2n-1$, so each of the sub-circuits above and below the waist must contain exactly $n-1$ operators. In the case when $n=1$, the top and bottom trees consist of single wires, and the circuit contains only the single operator that corresponds to the waist. For $n>1$, how should the $n-1$-operator sub-circuits above and below the waist be formed? The construction is recursive. A slice can by built by composing two smaller slices and then applying a function-preserving transformation to make the result into a WSO1 slice again, as shown in Figure 6. Applying this construction repeatedly results in slices with fan-out 2, in which the B and T sub-circuits are perfect binary trees. If these have depth k, then each has 2^k-1 operators. Then, the entire construction has $n=2^k+1$ inputs and size $2(2^k-1)+1=2^{k+1}-1$. Just to check the WSO property: $w+s=1+2^{k+1}-1=2^{k+1}=2(2^k+1)-2=2n-2$ as expected. Thus the widest slice whose waist stretches from level k to level $k+1$ has 2^k+1 inputs if fan-out is restricted to 2. Such a block is shown in Figure 7 for $k=4$. Note how, in the top tree, there is a path from every input to the rightmost output, while in the bottom tree there is a path from the leftmost input to each output.

2.4 Composing slices to make depth size optimal circuits

A WSO circuit whose waist is the same as its depth is depth size optimal (DSO). A DSO circuit can be built by composing WSO1 circuits with increasing *firstFork*, in the kind of pattern shown in Figure 8. For a circuit of depth d, d blocks are composed, with *firstFork* $0, 1, \ldots, d-1$. The composition of prefix circuits that we saw earlier (Figure 1(b)) is associative, and here we use it repeatedly. The simplest (and deepest) structure built by composing WSO1 blocks is the serial prefix network, in which each of the slices has 2 inputs. A network of depth d will then be constructed from d such 2-input blocks, giving a total of $2d-(d-1)=d+1$ inputs.

However, we can also make a DSO network of depth d by making sure that each of the d slices

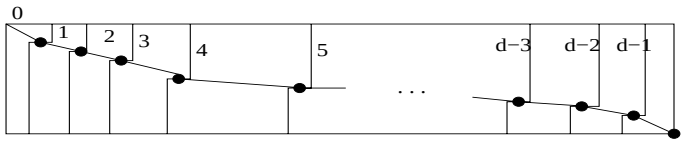

Figure 8: Composing slices to make a DSO circuit of depth d. The numbers shown give the *firstFork* of each of the d slices.

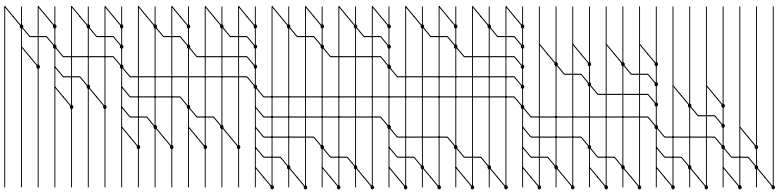

Figure 9: An example of Slices(2) with depth 9, built by composing 9 slices with fan-out 2.

is as wide as possible. This is the essence of our new construction, which we call Slices(f) for fan-out f. Here, we first consider Slices(2). For each composed slice, if the top tree has depth k, then the bottom tree must have depth at most $d-k-1$. Then, the smaller of k and $d-k-1$ determines the depth of the corresponding tree, and thus also the maximum possible width of the slice. Figure 9 shows the result for depth 9.

The total width of the parallel prefix circuit of depth d constructed in this way can be calculated by summing the widths of the WSO1 slices, making sure to take proper account of wires that are "lost" during composition.

$$\begin{aligned} \mathit{Width}(d) &= 1 + \sum_{k=0}^{d-1} 2^{min(k,d-k-1)} \\ &= 2(2^j) - 1, \quad d = 2j \\ &= 3(2^j) - 1, \quad d = 2j+1 \end{aligned}$$

Slices(2) is a pleasingly symmetrical construction. The reader might like to rotate Figure 9 through 180 degrees and to mentally slide the operators to the ends of the diagonal lines on which they appear. The result is the same circuit! This is partly because the construction of the maximal width WSO slice gives a symmetrical result. Also, we have seen that the sequence of depths that determine the widths of the d slices is $< min(k, d-k-1) | k : 0 \ldots d-1 >$, and reversing this sequence leaves it unchanged.

If one takes as the sequence of *firstFork* values the shorter $0, 1, \ldots, \lfloor d/2 \rfloor$, and also makes all trees as shallow as possible, one gets the basic Brent Kung construction [4] – compare Figures 3 and 9. For depth 9, the Brent Kung construction composes WSO1 blocks of width $2, 3, 5, 9$ and 17 to give an overall width of 32 (remembering that the composed blocks have a wire in common). We saw earlier that the Brent Kung construction is not quite DSO. It is, however, WSO because it is composed of WSO1 blocks.

Slices(2) can be seen as an alternative to the Brent Kung construction. For depth 9, Brent Kung has 32 inputs, as shown in Figure 3. Slices(2) encompasses 47 inputs for the same depth, see Figure 9. The corresponding numbers for depth 11 are 64 for Brent Kung and 95 for Slices(2). Both constructions are, however, rather deep compared to Sklansky and the minimum depth Ladner Fischer (although neither of those comes close to being DSO). Our next step is to explore the use of fan-out in the construction of reasonably shallow parallel prefix circuits that are still DSO (and thus small in size).

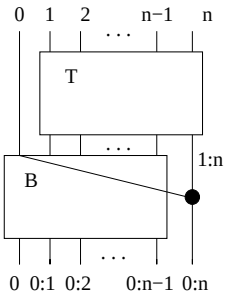

Figure 10: The form of a slice for fan-out greater than two.

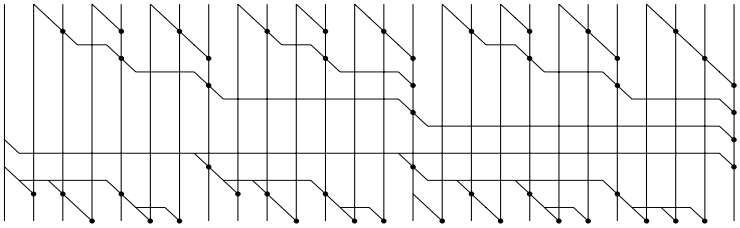

Figure 11: A slice that makes use of fan-out to allow the bottom tree to be shallower.

3 The Slices construction: fan-out > 2

Things become less beautiful but more interesting when we allow fan-out per level to increase. The overall construction of a DSO circuit is exactly the same as before – the composition of d slices with increasing *firstFork*. What changes is the method of constructing a maximal width slice for given constraints on the depth of the top and bottom trees.

3.1 Constructing a slice with top and bottom trees of given depth

The construction for fan-out $f > 2$ is similar to that used in the fan-out 2 case, except that the lower tree can now have its first fork on its first wire at the same level as the fork in the waist, causing the whole tree to move up one level as shown in Figure 10. At the point of the fork in the waist, the B tree can have fan-out of $f - 1$, since the fan-out to the operator in the waist counts as one of the available fan-outs. At all other fork points in B, there can be fan-out of f.

It turns out that increasing allowed fan-out above two makes no difference to the fan-out of the top tree, since its main purpose is to produce a single output (the rightmost one) that depends on all of the inputs. Additional fan-out does not assist with this goal. On the other hand, the bottom tree aims to spread a single input to each of the outputs, so fan-out can most definitely play a rôle here. For example, Figure 11 shows a slice with fan-out 4 and with depths 5 and 3 for the top and bottom trees. Note how the bottom tree is squashed up against the wire that goes across the waist.

For a given depth and fan-out, we want to construct the widest possible DSO parallel prefix circuit. We will construct such a DSO circuit from slices, so the problem reduces to how to construct a widest possible slice with top tree depth t and bottom tree depth b, for a given fan-out $f > 2$. What we want is a pair of matching trees, one with fan-out 2 and depth at most t, and one with fan-out f and depth at most b (and with the bottom tree including the waist as illustrated in Figure 11). Again, a recursive construction is used.

First, consider the simpler problem of how to make matching top and bottom trees in the absence of any constraints on the fan-out. In that case, the recursive construction is illustrated

in Figure 12. A base case is reached when either the b parameter (for the depth of the bottom tree) or the t parameter (for the depth of the top tree) is zero. In both cases, the resulting tree is just a single wire, without any operators. For the recursive case, the entire top tree must have depth t, and then the two recursive calls must each have depth $t-1$. In the bottom tree, the left hand recursive call must have the same depth as the entire tree (that is b), while the right hand recursive call must decrease the b depth parameter by one. The recursive calls are shown in the diagram as boxes containing the appropriate parameters. The two tree constructions have a similar recursive pattern, and both have the same number of inputs and operators. Let $B1$ be the number of input wires to the bottom tree. It is computed by the following integer recurrence:

$$\begin{aligned} B1(0,t) &= 1 \\ B1(b,0) &= 1 \\ B1(b,t) &= B1(b,t-1) + B1(b-1,t-1) \end{aligned}$$

This recurrence is closely related to Pascal's 6th identity, which states that

$$\sum_{k=0}^{l} \binom{n}{k} = \sum_{k=0}^{l} \binom{n-1}{k} + \sum_{k=0}^{l-1} \binom{n-1}{k}$$

This leads directly to the solution

$$B1(b,t) = \sum_{i=0}^{b} \binom{t}{i} \tag{1}$$

The next step towards taking account of fan-out is to unroll the recursion in the above construction, as shown in Figure 13. For the bottom tree, this distinguishes clearly between the top level of the tree and the remaining levels. Note that the top level of the bottom tree has fan-out $t+2$, when one includes the waist (which is shown shaded).

Finally, we introduce an extra parameter f for the allowed fan-out per level. Figure 14 shows the construction of matching top and bottom trees, T' and B', with fan-out f at each level of the bottom tree (provided t is large enough). When the fan-out to the waist (shown shaded) is added, this gives fan-out $f+1$ at the top of the bottom tree, which is too high. Our solution is to delete parts of the construction, at the right hand side, as indicated by dotted lines; we call the result a *slice*. The resulting top tree, T, has depth $t-1$, while the bottom tree, B, has depth b, the same as that of B'. This construction of *slice* applies when $f \leq t+1$. If $f > t+1$ (as will happen at the leaves of the recursion), it is t that determines the allowed fan-out, and we revert to the previous case, shown in Figure 13. This will be reflected in the recurrence for the number of inputs to a slice that we develop later.

Figure 15 shows a concrete example of a slice constructed in this way. Note that the first fork is the depth of the top tree, in this case 6.

The values of both b and t constrain both trees. It can happen, for instance, that the depth of the top tree ends up being less than t because the value of b is small, or vice versa. As a concrete example, if $t = 1$ then the depth of the resulting bottom tree will be 1, independent of the value of b, provided $b > 0$.

3.2 The correctness of this method of constructing a slice

The *slice'* construction shown in Figure 14, including the B', T' and waist, clearly has waist one. It can be shown to be a WSO parallel prefix circuit by a transformation that replaces the serial prefix in the top tree, and the top level of the bottom tree plus waist by a single serial prefix. The result is then the composition of f smaller *slice'* circuits; it is a WSO parallel prefix circuit, because it is the composition of WSO1 blocks. The transformation (and its inverse) preserves both the WSO property and circuit behaviour, so our original circuit is also a WSO parallel prefix circuit. The smaller *slice* construction can be obtained by composing $f-1$ (rather than f) smaller *slice'* circuits, and then applying a transformation to the waist similar to the inverse of that described

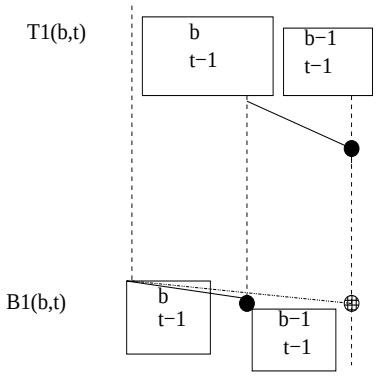

Figure 12: Recursive construction of matching top and bottom trees. The parameters b and t are the depths of the bottom and top trees. The only constraint on fan-out is the allowed depth of the top tree. The operator on the waist is shown shaded.

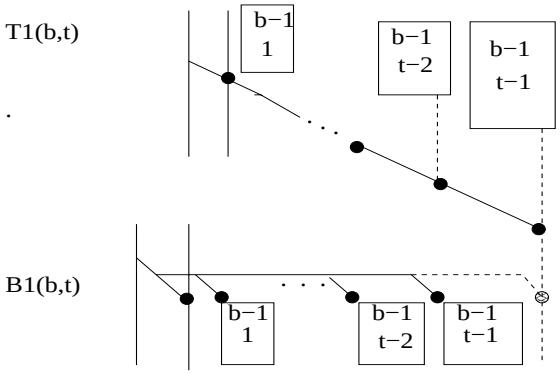

Figure 13: Unrolling the recursion in the construction of top and bottom trees.

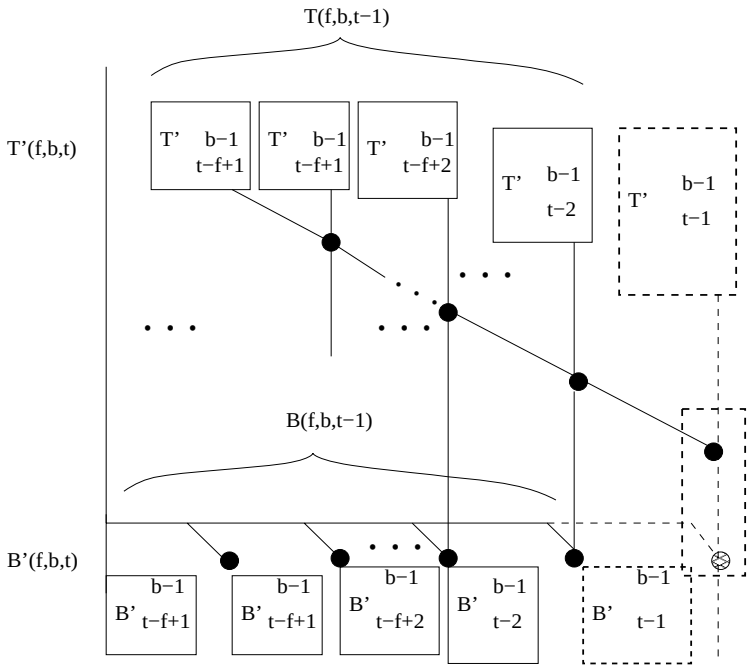

Figure 14: The construction of the *slice'*, with matching top and bottom trees T' and B', allowing fan-out f at each level of the bottom tree (provided t is sufficiently large). Because *slice'* has fan-out $f+1$ at the top level of B', we delete parts of the construction, at the right hand side, as indicated by dotted lines, to give *slice*. The resulting top tree, T, has depth $t-1$, while the bottom tree, B, has depth b, the same as that of B'

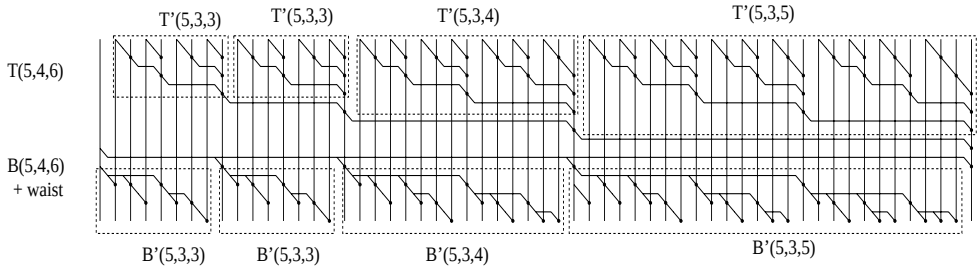

Figure 15: A slice with fan-out 5, bottom tree depth 4 and top tree depth 6, showing the recursive construction.

above. The result is a WSO1 parallel prefix circuit because it has the same behaviour and number of operators as the composition of $f-1$ WSO1 parallel prefix circuits. The fan-out 2 case that we considered in section 2 is an instance of this more general one.

3.3 Does the slice encompass the maximum possible number of inputs?

Although we have constructed the slice to have as great a width as possible for the given fan-out and bottom and top tree depths, we do not yet have a satisfactory proof that this is the case. This is left to future work.

3.4 Analysing the width of a slice

To calculate the maximum number of inputs that Slices(f) can process for a given depth, we must first calculate the width of each individual slice.

We often need to sum over sequences of the form $1, 2, \ldots, g-1, g, g$ – that is monotonically increasing sequences with two copies of the last element. We will use the notation $\sum_{i=1}^{g,g}$ when summing over such sequences.

Consider the recursive construction outlined in Figure 14, for $f \leq t+1$ & $b > 1$ & $t > 0$. Let $W(f,b,t)$ be the number of inputs in the B and T constructions, and $W'(f,b,t)$ the number of inputs in the B' and T' constructions. B' is made from f smaller copies of B', giving

$$W'(f,b,t) = \sum_{k=1}^{f-1,f-1} W'(f,b-1,t-k) \tag{2}$$

Since B is obtained by deleting one copy of B', the relationship between W and W' is as follows:

$$W(f,b,t) = W'(f,b,t+1) - W'(f,b-1,t) \tag{3}$$

Thus, we can find a recurrence for W in terms of itself:

$$\begin{aligned}
W(f,b,t) &= W'(f,b,t+1) - W'(f,b-1,t) && \text{[eq. 3]}\\
&= \sum_{k=1}^{f-1,f-1} W'(f,b-1,t+1-k) - \sum_{k=1}^{f-1,f-1} W'(f,b-2,t-k) && \text{[eq. 2]}\\
&= \sum_{k=1}^{f-1,f-1} \left(W'(f,b-1,t+1-k) - W'(f,b-2,t-k)\right) && \text{[summation]}\\
&= \sum_{k=1}^{f-1,f-1} W(f,b-1,t-k) && \text{[eq. 3]}
\end{aligned}$$

The full recurrence for W is

$$\begin{aligned}
W(f,b,t) &= 1, && b = 0 \vee t = 0 && (4)\\
&= W(f,b,t-1) + W(f,b-1,t-1), && f > t+1 && (5)\\
&= f-1, && f \leq t+1 \;\&\; b = 1 && (6)\\
&= \sum_{k=1}^{f-1,f-1} W(f,b-1,t-k), && f \leq t+1 \;\&\; b > 1 && (7)
\end{aligned}$$

The first equation gives the base cases: the tree is a single wire if either the top or the bottom tree should be of depth zero. The second equation covers the case when the fan-out is high compared to the top tree depth, and follows the construction shown in Figure 12. It can also be shown by induction that

$$W(f,b,t) = 1 + \sum_{i=1}^{t} W(f,b-1,t-i), \quad f > t+1 \tag{8}$$

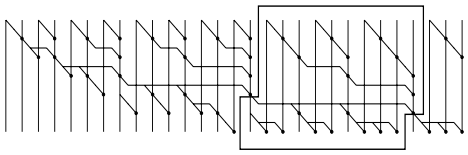

Figure 16: Slices(4) with depth 6. The outlined slice has a top tree of depth 4 and a bottom tree of depth 2.

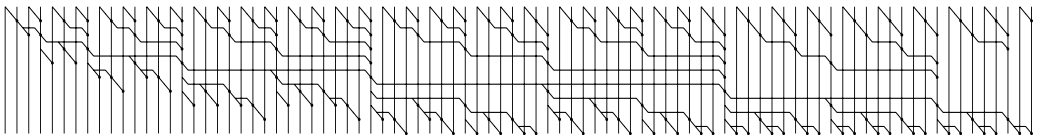

Figure 17: Slices(3) with 88 inputs and depth 9. Compare with Figure 9, which has the same depth but fan-out 2.

Equations 6 and 7 give the base case and step to calculate the number of inputs to our new fan-out-constrained construction, shown in Figure 14.

3.5 Building DSO circuits Slices(f), for fan-out $f > 2$

As before, to build a DSO circuit of depth d, slices with *firstFork* ranging from 0 to $d-1$ are composed. To emphasise the effect of increasing fan-out, we show three further examples. In Figure 9, we saw Slices(2), with depth 9 and 47 inputs. Increasing the fan-out to 3 and 4 allows us to build DSO circuits with the same depth, but with 88 and 114 inputs respectively, as shown in Figures 17 and 18.

If we again choose depth 9, but make the fan-out be 10, then the top trees in the resulting composition of slices have depths $0, 1, 2, \ldots, 8$. This means that in each of the slices, it is the case that $f > t + 1$, so that we revert to the recursive construction that does not take account of fan-out (see Figure 12). This is the construction described by Zhu et al [27], and it is shown in Figure 19. In general, the instance of our construction in which the fan-out is set to be $d + 1$ is essentially the same as the Zhu network. We will return to this point in section 5.2.2.

3.6 Analysing the width of the Slices(f) construction for a given depth

The maximum size DSO parallel prefix circuit for a given depth and fan-out is calculated based on the widths of its d slices:

$$M(f, d) = 1 + \sum_{i=0}^{d-1} W(f, d-i, i) \tag{9}$$

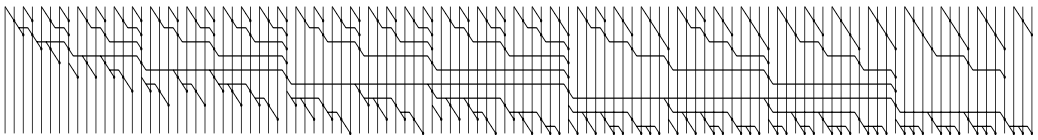

Figure 18: Slices(4) with depth 9. 114 inputs can be processed.

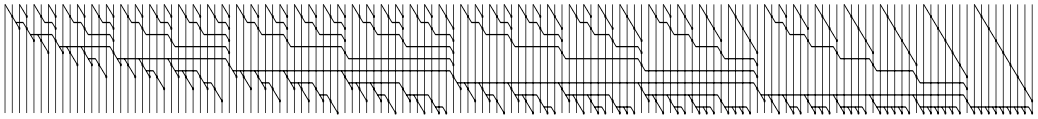

Figure 19: Slices(10) with depth 9. 143 inputs can be processed (see Table 1 in [27]).

Interestingly, it is possible to give a recursive definition of M. We first consider the case of $f > d$.

$$
\begin{aligned}
M(f,d) &= 1 + \sum_{i=0}^{d-1} W(f, d-i, i) \\
&= 1 + 1 + \sum_{i=1}^{d-2} W(f, d-i, i) + d && [W(f,d,0) = 1, W(f,1,d-1) = d] \\
&= 2 + d + \sum_{i=1}^{d-2} W(f, d-i, i-1) + \sum_{i=1}^{d-2} W(f, d-i-1, i-1) && [\text{eq. 5}] \\
&= 2 + d + \sum_{i=0}^{d-3} W(f, d-1-i, i) + \sum_{i=0}^{d-3} W(f, d-2-i, i) && [\text{change of index}] \\
&= 1 + 1 + \sum_{i=0}^{d-2} W(f, d-1-i, i) + 1 + \sum_{i=0}^{d-3} W(f, d-2-i, i) && [W(f,1,d-2) = d-1] \\
&= 1 + M(f, d-1) + M(f, d-2) && [\text{eq. 9}]
\end{aligned}
$$

For $f <= d$:

$$
\begin{aligned}
&M(f,d) \\
&\quad = 1 + \sum_{i=0}^{d-1} W(f, d-i, i) \\
&\quad = 1 + W(f,d,0) + \sum_{i=1}^{f-2} W(f, d-i, i) + \sum_{i=f-1}^{d-2} W(f, d-i, i) + W(f, 1, d-1) \\
&\quad = 1 + f + \sum_{i=1}^{f-2} (1 + \sum_{j=1}^{i} W(f, d-i-1, i-j)) + \sum_{i=f-1}^{d-2} \sum_{j=1}^{f-1,f-1} W(f, d-i-1, i-j) && [\text{eqs. 4,6, 7, 8}] \\
&\quad = 1 + f + \sum_{j=1}^{f-2} (1 + \sum_{i=0}^{f-2-j} W(f, d-i-j-1, i)) + \sum_{j=1}^{f-1,f-1} \sum_{i=f-1-j}^{d-2-j} W(f, d-i-j-1, i) \\
&\quad = 1 + f + \sum_{j=1}^{f-2} (1 + \sum_{i=0}^{d-2-j} W(f, d-1-j-i, i)) \; + \; 2 \times \sum_{i=0}^{d-2-(f-1)} W(f, d-i-(f-1)-1, i) \\
&\quad = 1 + f + \sum_{j=1}^{f-2} M(f, d-1-j) \; + \; 2 \times M(f, d-1-(f-1)) - 2 && [\text{eq. 9}] \\
&\quad = f - 1 + \sum_{j=1}^{f-1,f-1} M(f, d-1-j)
\end{aligned}
$$

Thus, the recurrences for M are very similar to those for W. To summarise, we have

$$M(f,0) = 1 \tag{10}$$
$$M(f,1) = 2 \tag{11}$$
$$M(f,d) = 1 + M(f,d-1) + M(f,d-2) \qquad f > d \tag{12}$$
$$= f - 1 + \sum_{j=1}^{f-1,f-1} M(f,d-1-j) \qquad f \leq d \tag{13}$$

And what is the point? The point is that given f and d, these simple recurrences tell you what width of DSO parallel prefix circuit you can build. As far as we know, nobody has been able to do this before.

4 Cropping Slices so as to process a given number of inputs

We have shown how to construct a maximum width circuit for a given fan-out and depth. But it is more usual that the number of inputs is decided first. A slice can be made narrower by replacing each of a pair of matching sub-trees in the structure by single wires. Think of replacing top trees by their rightmost wire (and no operators) and bottom trees by their leftmost wire. When we delete a pair of matching sub-trees, k wires are removed, and for each tree, k operators. The waist is unaffected, and remains at one. This means that the constraint on waist and size: $w + s = 2n - 2$ remains intact. Thus, the slice remains WSO1 even after cropping. It is still a parallel prefix network because it can still be transformed into the composition of $f-1$ blocks that are themselves parallel prefix networks, as before. Some of those blocks have been made narrower by the same process, so an inductive proof is required.

Then, to make an instance of Slices narrower, one can crop any of its slices in this way. The waist of the circuit, which stretches from the top left corner (the first fork) to the bottom right, through a number of operators, is unaffected by this cropping, and the depth of the resulting circuit therefore remains unchanged. Then, because the resulting circuit is WSO, being composed of WSO blocks, and has waist equal to its depth, it is also DSO.

To illustrate this process, Figures 20 to 22 show three networks that have depth 8 and fan-out 4. The first is the original (maximum width for that fan-out and depth) construction with 72 inputs. The three sub-trees marked in that diagram (and their corresponding bottom trees) are replaced by single wires, to give the 64 input network shown in Figure 21. Removing further sub-trees in this way gives, for example, the 57 input network in Figure 22. Here, we have chosen always to replace a rightmost sub-tree by a wire, so that deletion proceeds from right to left. Other choices would also be possible, for instance one that starts shrinking the slices from the left. If aiming for implementation as a circuit, one might want to start cropping in the middle, because this would reduce the length of the longest wire in the waist of the circuit. Figure 23 shows such a network, again for 57 inputs.

In the limit, one can replace all the sub-trees with single wires, leaving the waist of the circuit intact, but removing all else. At that point, the size is the same as the depth, d, and the number of inputs is $d + 1$. This gives the serial prefix network. Further cropping is not possible as this would reduce the depth.

Note that simply cropping the parallel prefix network by deleting wires and associated operators on the right would not be guaranteed to give a result that is DSO. For example, deleting the three rightmost wires of the 72 input circuit in Figure 20 would give a circuit whose *waist*, w, is one smaller than its depth, d. But we know that $w + s \geq 2n - 2$, which means that in this case, $d + s > 2n - 2$, so the circuit is not DSO. The cropping method described here preserves the DSO property.

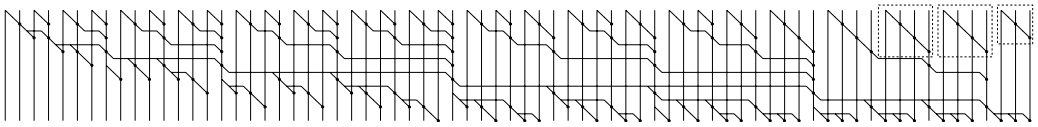

Figure 20: A widest possible DSO parallel prefix circuit with depth 8, fan-out 4 and 72 inputs. The marked sub-trees (and their matching bottom trees) will each be replaced by single wires, to give the following 64 bit construction.

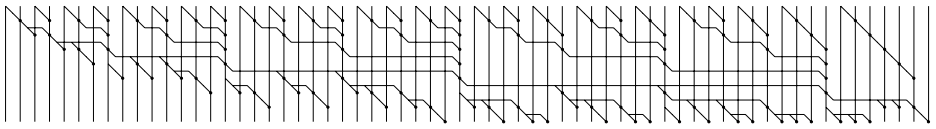

Figure 21: A 64 input DSO network constructed by cropping the full-width network above.

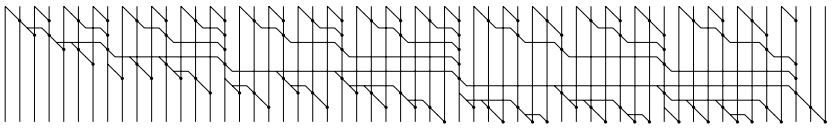

Figure 22: Continuing the cropping process, which narrows the individual slices. The example shown has 57 inputs. The two rightmost slices have been shrunk to width 2.

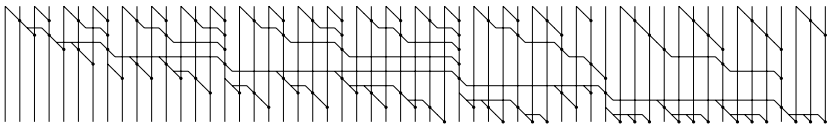

Figure 23: Another example of a 57 input instance of Slices, this time with cropping of the wider slices in the middle, which reduces the length of the longest wire along the waist.

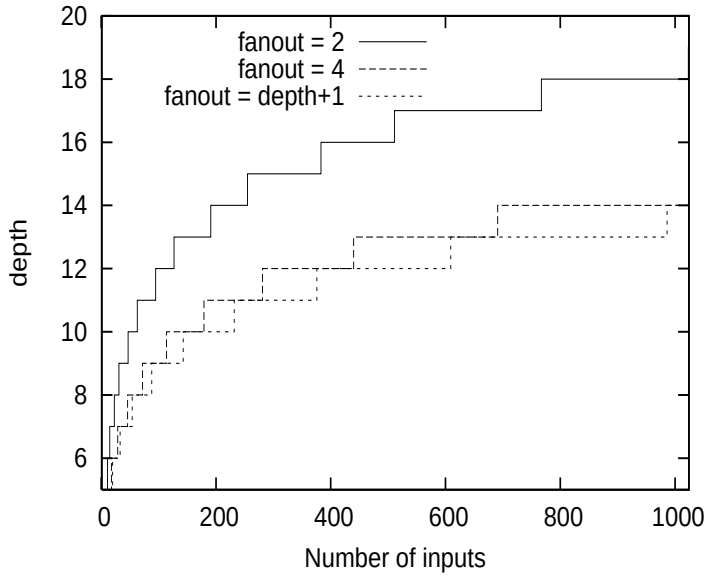

Figure 24: The effect of varying fan-out in the Slices construction.

5 Analysis of results

We present a first analysis of our results. Much remains to be done.

5.1 The effect of varying fan-out in the Slices construction

Figure 24 plots depth against number of inputs for the Slices construction with three different fan-outs (2,4 and as large as the construction will allow, that is depth +1).

Here, note how close the fan-out 4 is to the "unconstrained" one. For example, for $n = 233..281$, fan-out 4 gives the same depth (11) as the unconstrained fan-out (which must be 12, that is depth +1). And in the range shown, and indeed up to 4163 inputs, the fan-out 4 graph is within depth 1 of the unconstrained one. Thus, fan-out 4 can be argued for from an algorithmic point of view, and not only from an electrical one. This is fortunate!

5.2 Comparison with other parallel prefix networks

5.2.1 The Ladner Fischer Constructions

First, let us look at some concrete examples, and then attempt a more general comparison between Slices and Ladner Fischer. For 32 inputs, $P_0(32)$ has fan-out 17, and size 74, as we saw in Figure 4 [13]. Our construction cannot produce a network with 32 inputs and minimum depth 5. Allowing the depth to increase by one, $P_1(32)$ has fan-out 9 and size 62. If, in the Slices construction, we restrict depth to be 6 and number of inputs to be 32, we find that we must have fan-out of at least 6, since $M(5, 6) = 31$ and $M(6, 6) = 32$. The resulting network, which has size 56, is shown in Figure 25. So, for fixed depth 6 and number of inputs 32, both fan-out and size are noticeably improved compared to Ladner Fischer.

For 128 inputs, the Ladner Fischer construction gives depth 7, size 369, depth 8, size 295 and depth 9, size 264 networks, with the latter having fan-out 19. Our construction can achieve depth 9 at best. The resulting network has size 245 and fan-out of only 5, as shown in Figure 26. Again, there is a considerable improvement in fan-out and size for the same depth and number of inputs, compared to Ladner Fischer.

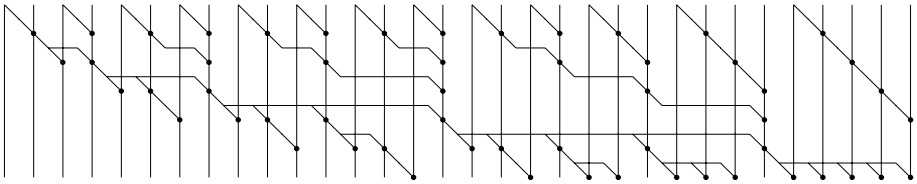

Figure 25: A 32 input Slices network with depth 6 and fan-out 6.

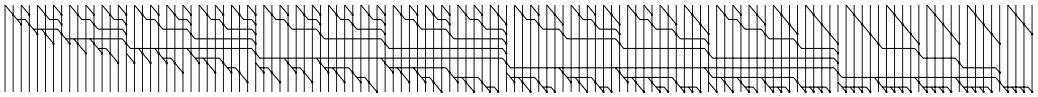

Figure 26: A 128 input Slices network with depth 9 and fan-out 5.

Now let us try to make a more general comparison between Slices(4) and the Ladner Fischer constructions. First, in Figure 27, we plot depth against maximum number of inputs for the shallowest Ladner Fischer construction (which has minimum depth), Brent Kung and Slices(4). Slices(4) sits squarely in the middle. It would be interesting to try to analyse this more formally. Here, we make a first observation. Much remains to be done.

On the other hand, when we plot number of operators against number of inputs, in Figure 28, we find that Slices(4) has *much fewer* operators than the shallowest Ladner Fischer (which in turn has considerably fewer operators than Sklansky), while having slightly fewer than Brent Kung. Note, also, that the shallowest Ladner Fischer has fan-out $\lfloor n/2 \rfloor + 1$, compared to 4 for Slices here, and 2 for Brent Kung. Remember, though, that Ladner Fischer has minimum depth.

Generally, which of these three options one should choose depends on the relative importance of depth, size and fan-out. But still, it is clear that Slices adds an interesting option to the existing standard constructions, particularly if one wants low fan-out, small size but relatively low depth. These circumstances might arise, for example, in circuit designs aimed at low power rather than raw speed. However, it should be remembered that our analyses are, at this stage, rather abstract.

5.2.2 The network of Zhu et al

The construction of the zero-deficiency parallel prefix network $Z(d)$, which has the maximum possible width for depth d, appears to be exactly the same as ours for the case in which fan-out is $d + 1$ [27, 28]. This is confirmed by examining the diagrams (e.g. on p. 13 of [28]) and the given construction, and by comparing the figures for maximum width for a given depth given. For example, for depth 8 fan-out 9, both constructions give width 88, while for depth 9, fan-out 10, both give width 143 (see Figure 19). If we assume that $f > d$, our calculation of the width for depth d of Slices (equation 12), which we will here call $S(d)$, can be rewritten as

$$\begin{aligned} S(0) &= 1 \\ S(1) &= 2 \\ S(d) &= 1 + S(d-2) + S(d-1), d > 1 \end{aligned}$$

while Zhu et al have derived, for $d \geq 1$, the closed form

$$N_Z(d) = F(d+3) - 1$$

where $F(r)$ denotes the r^{th} Fibonacci number with $F(1) = F(2) = 1$, and $F(k+2) = F(k+1) + F(k), k \geq 0$. It is an easy induction to show that $S(d) = N_Z(d)$ for $d \geq 1$. For the base case, $S(1) = 2$ and $N_Z(1) = F(4) - 1 = 2$.

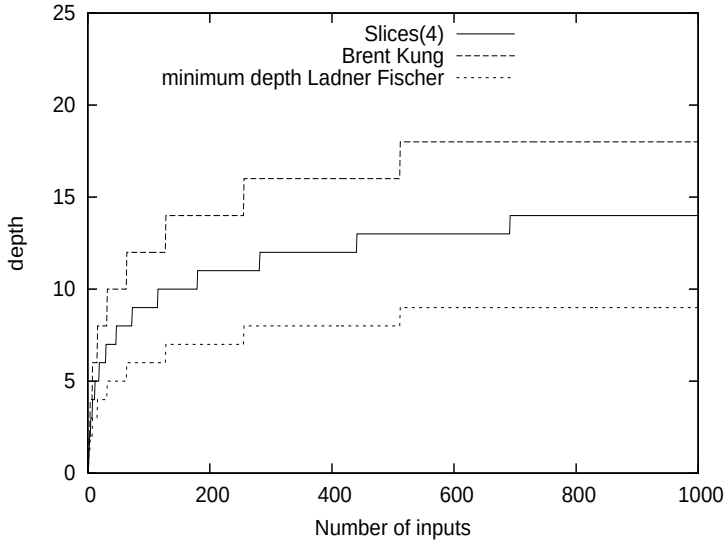

Figure 27: Depth for a given number of inputs, comparing the shallowest Ladner Fischer construction and Brent Kung with Slices(4).

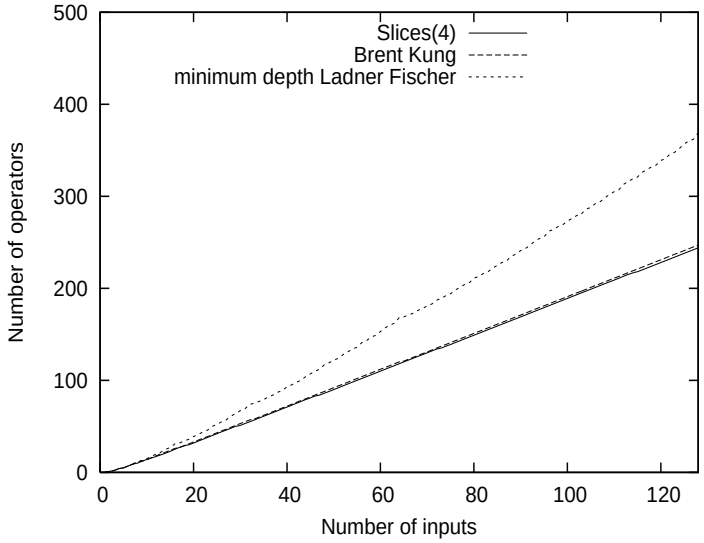

Figure 28: Number of operators for a given number of inputs, comparing the shallowest Ladner Fischer construction and Brent Kung with Slices(4).

For the step,

$$\begin{aligned} S(d+1) &= 1 + S(d-1) + S(d) \\ &= 1 + N_Z(d-1) + N_Z(d) \text{ by ind. hyp.} \\ &= 1 + F(d+2) - 1 + F(d+3) - 1 \\ &= F(d+4) - 1 \\ &= N_Z(d+1) \end{aligned}$$

When it comes to the width of an individual slice in the case in which fan-out is not constrained, the link that we found to Pascal's 6^{th} identity establishes equation 1 (giving the width of the trees in a slice with given bottom and top tree depths) more simply than the proof by Zhu et al [28]. Indeed, this link is very much in the spirit of their paper!

What we have done that has not yet been achieved by Zhu et al is to systematically construct DSO networks with constrained fan-out, $f \leq d$, and to analyse the width of the resulting networks in equations 10 to 13. Working independently, Zhu et al took a first step in this direction by explicitly deleting parts of the depth 8 fan-out 9 construction with 88 inputs to give a fan-out 4 network that is (we think) identical to our 72 input fan-out 4 construction. Our contribution has been to build limited fan-out into the Slices construction from the start.

In generalising their construction, Zhu et al propose a method of generating DSO circuits for arbitrary (n, d) pairs. In the case when the depth is the minimum possible for that number of inputs, they propose simply removing most significant inputs and outputs (and associated operators) from the maximum width construction. This, as we pointed out above, does not necessarily give a DSO result. This problem is probably what they refer to when they point out that "Care must be taken to make small adjustments to the new most significant columns after discarding.". Our approach in which matching sub-trees are replaced by wires seems more systematic, particularly since it carefully preserves the waist of the circuit. Note, incidentally, that what we call waist is called *ridge* by Zhu et al.

5.2.3 Comparing to previous bounded fan-out DSO networks

It is also interesting to compare with the fan-out 4 networks created by Lin and his co-workers. A recent paper by Lin and Su proposes a new construction called $SU4$ for DSO parallel prefix networks with fan-out 4 [16]. This is the culmination of a whole series of such constructions [14, 15] and is superior to the others in the series. Thus, we compare only with this latest construction. $SU4$ is built from 6 different types of parallel prefix circuits, which are used as building blocks. Our construction has only one type of building block, the slice, with its matching trees. Thus, our construction is conceptually much simpler, and consequently easier to analyse. In addition, the Slices(4) network is superior to $SU4$ in that it encompasses a greater number of inputs for all depths from 10 and upwards. For larger depths, this superiority is marked. Figure 29 shows the number of inputs and depth for the two constructions.

5.2.4 Comparison with circuit synthesis-oriented approaches

In the early 1990s, Fishburn developed a tool called LATTIS that used heuristic procedures for performance optimisation of mapped combinational logic [8, 7]. Interestingly, in the earlier paper, parallel prefix is taken as a case study and parallel prefix networks are developed by greedily applying a depth-decreasing heuristic to a serial prefix network. The result is compared to Ladner Fischer. For instance, for 32 inputs, and depth 8, LATTIS produces a DSO network of 54 operators with fan-out 5, while $P_3(32)$, the Ladner Fischer construction of depth 3 greater than minimum depth, has 57 operators and fan-out 6. It is interesting to note that LATTIS produces a network that is a composition of slices, but that it does not choose exactly the right slice widths to give the minimum possible depth for such a construction. Fishburn points out that his heuristic approach produces DSO circuits in the depth range $2\log_2 n - 2$ to $n - 1$, for n inputs, and the above example is at the lower limit of this range. Our non-heuristic approach (which is, of course, much more

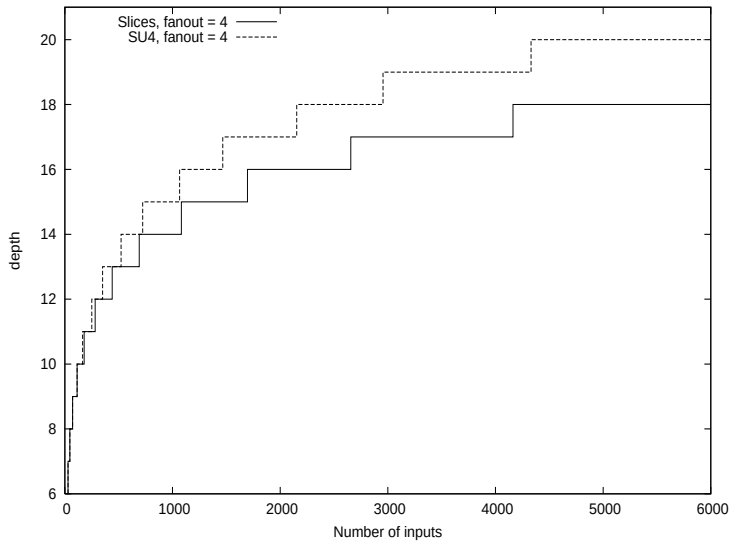

Figure 29: Comparing $SU4$ with Slices(4). Slices(4) copes with more inputs already at depth 10 (179 vs. 165). At depth 20, the maximum number of inputs is 15997 for Slices(4) and 5931 for $SU4$.

specialised) produces DSO networks also for smaller depths. For instance, in Figure 25, we saw a DSO network of depth 6 for 32 inputs (and with fan-out only 6). On the other hand, Fishburn's approach took account both of input arrival time and desired output time (using a unit-delay model for gates). We are also very interested in pursuing this direction.

In a non-heuristic approach, Zimmermann has studied the generation of optimal parallel prefix networks, including taking account of non-uniform input and output signal arrival times [30, 29]. For uniform input signal arrival times, the results seem quite similar to ours. For instance, a DSO network with 32 inputs and depth 6 is shown, and indeed it too is a composition of slices [30]. Its fan-out is 7, one greater than that in our construction. We have found Zimmermann's work inspiring, and hope to return to a more serious comparison when we have worked with non-uniform input signal arrival times.

6 Future Work

We became interested in parallel prefix networks because we want to solve real problems in circuit design. A referee of an early (admittedly incoherent) attempt to explain these ideas claimed that we (like everyone else) are still at the stage that Weinberger was at in '59, trying to "figure a good tradeoff between delay and area/size/energy". Trying to understand this tradeoff is indeed one of our goals, and we feel that we have actually done something useful towards this end. We have understood how to produce parallel prefix networks with the smallest possible number of operators for given constraints on fan-out, depth and number of inputs. We assert that understanding how to make low-fan-out parallel prefix circuits that do not have many long wires is going to be important in practice. These constructions are a first step. The next step is to take account of wires and their effects, and this is a topic that we are pursuing actively [1].

It remains to be seen if these constructions will have desirable properties when implemented as circuits, although a first implementation (by undergraduate students at Chalmers) of a 64-bit adder based on these ideas had 15% lower power consumption than a reference Han-Carlson implementation, though at the expense of being 15% slower. It is true that the DSO condition only counts number of operators, and ignores real physical effects, but this does *not* imply that

the resulting networks perform poorly as circuits, and first indications are promising. Many more experiments are needed. Our intention, though, is not to implement these structures directly, but to use the insights gained in developing these constructions to do synthesis of parallel prefix networks that are adapted to the delay profile of their inputs.

Note that parallel prefix circuits appear in many guises, so it makes sense to start with the general case, as we have done here, without any assumptions about the final application area. Now, however, further work is needed to specialise these results to the important special case of binary addition (taking account of generate and propagate signals). This has *deliberately* not been done yet. The more abstract analysis is necessary as a base on which to build more specialised ones. We will both use our Wired language to generate real adder layouts, and the combination of Wired with standard modelling methods, including methods of estimation based on comparing energy-delay behaviour [20] and power macromodelling [5].

On the more theoretical side, much work also remains to be done. We have designed the Slices construction to encompass as many inputs as possible for a given depth, but we do not yet have a satisfactory proof that we have indeed succeeded! This must be our next step.

Next, having studied the effect of fan-out in the region in which DSO networks exist, it would be interesting to move on to consider fan-out in the more chaotic region in which tight depth constraints prevent them from existing, and size grows quickly. Beyond these questions about how size, number of operators and fan-out relate to each other, more theoretical work needs to be done to take account of *area* rather than number of operators. The classic paper by Sugla and Carlson on extreme area-time trade-offs in VLSI points the way here [24].

Parallel prefix in the context of parallel programming rather than circuits is yet another area that we should consider.

7 Conclusion

The Slices construction is interesting because it provides a new family of relatively shallow parallel prefix networks that are also small. Previous results are improved upon in a quantitative sense. However, perhaps the most important contribution is some new insight into how depth, size and fan-out can be traded off against each other. Much remains to be done, but these are important first steps! We hope that some of our readers will join us in this exploration, and would welcome comments and feed-back on this technical report.

Acknowledgements

This work is funded, in part, by the Swedish Research Council, Vetenskapsrådet, and by the Semiconductor Research Corporation in an Intel-custom project. Thanks to Burton Smith for pointing us to some relevant papers on the relationship between parallel prefix and loop parallelization. Thanks also to Emil Axelsson, both for spotting errors and for understanding this obsession with parallel prefix. This work grew out of a visit by Sheeran to Intel Strategic CAD Labs, where many of the researchers she spoke to were designing parallel prefix circuits.

References

[1] E. Axelsson, K. Claessen, and M. Sheeran. Wired: wire-aware circuit design. In *Correct Hardware Design and Verification Methods, CHARME*, volume 3725 of *Lecture Notes in Computer Science*. Springer, 2005.

[2] O. J. Bedrij. Carry-select adder. *IRE Transactions on Electronic Computers*, pages 340–346, 1962.

[3] G.E. Blelloch. Scans as Primitive Parallel Operations. In *Proc. of the International Conference on Parallel Processing*, pages 355–362, August 1987.

[4] R.P. Brent and H.T. Kung. A regular layout for parallel adders. *IEEE Transactions on Computers*, C-31:260–264, 1982.

[5] M. Q. Do, M. Drazdziulis, P. Larsson-Edefors, and L. Bengtsson. Parameterizable architecture-level sram power model using circuit-simulation backend for leakage calibration. In *Intl Symp. on Quality Electronic Design*, March 2006.

[6] Faith E. Fich. New bounds for parallel prefix circuits. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*. ACM Press, 1983.

[7] J. P. Fishburn. Lattis: an iterative speedup heuristic for mapped logic. In *DAC '92: Proceedings of the 29th ACM/IEEE conference on Design automation*. IEEE Computer Society Press, 1992.

[8] John P. Fishburn. A depth-decreasing heuristic for combinational logic: or how to convert a ripple-carry adder into a carry-lookahead adder or anything in-between. In *DAC '90: Proceedings of the 27th ACM/IEEE conference on Design automation*. ACM Press, 1990.

[9] Ralf Hinze. An algebra of scans. In *Mathematics of Program Construction*, volume 3125 of *Lecture Notes in Computer Science*, pages 186–210. Springer, 2004.

[10] Z. Huang and M. Ercegovac. Effect of Wire Delay on the Design of Prefix Adders in Deep-Submicron Technology. In *Proc. 34th Asilomar Conf.* IEEE, 2000.

[11] D.B.G. Kilburn T., Edwards and D Aspinall. Parallel addition in digital computers: A new fast carry circuit. *Proceedings of IEE*, 106, pt B, 1959.

[12] P.M. Kogge and H.S. Stone. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. *IEEE Transactions on Computers*, C-22(8):786–793, 1973.

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

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

[15] Y.-C. Lin, Y.-H Hsu, and C.-K. Liu. Constructing H4, a Fast Depth-Size Optimal Parallel Prefix Circuit. *The Journal of Supercomputing*, 24(3):279–304, 2003.

[16] Y.-C. Lin and C.-Y. Su. Faster optimal parallel prefix circuits: New algorithmic construction. *Journal of Parallel and Distributed Computing*, 65(12), 2005.

[17] M. Nadler. A high-speed electronic arithmetic unit for automatic computing machines. *Acta Techn.*, 16:464–478, 1956.

[18] Alexandru Nicolau and Haigeng Wang. Optimal schedules for parallel prefix computation with bounded resources. In *PPOPP '91: Proceedings of the third ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM Press, 1991.

[19] Y.P. Ofman. The algorithmic complexity of discrete functions. *Sov. Phys. Dokl.*, 7:589–591, 1963.

[20] V. Oklobdzija, B. R. Zeydel, S. Mathew H. Q. Dao, and R. Krishnamurthy. Comparison of high-performance vlsi adders in energy-delay space. *IEEE Transaction on VLSI Systems*, 13(6), June 2005.

[21] Nicholas Pippenger. The complexity of computations by networks. *IBM J. Res. Dev.*, 31(2):235–243, 1987.

[22] J. Sklansky. Conditional-sum addition logic. *IRE Trans. Electron. Comput.*, EC-9:226–231, 1960.

[23] Marc Snir. Depth-size trade-offs for parallel prefix computation. *J. Algorithms*, 7(2):185–201, 1986.

[24] Binay Sugla and David A. Carlson. Extreme area-time tradeoffs in vlsi. *IEEE Trans. Computers*, 39(2), 1990.

[25] Haigeng Wang, Alexandru Nicolau, and Kai-Yeng S. Siu. The strict time lower bound and optimal schedules for parallel prefix with resource constraints. *IEEE Trans. Comput.*, 45(11), 1996.

[26] A. Weinberger and J.L. Smith. A logic for high-speed addition. *National Bureau of Standards, Circ. 591*, pages 3–12, 1958.

[27] Haikun Zhu, Chung-Kuan Cheng, and Ronald Graham. Constructing Zero-deficiency Parallel Prefix Adder of Minimum Depth. In *Asia South Pacific Design Automation Conference, ASP-DAC*, pages 883–888. IEEE, 2005.

[28] Haikun Zhu, Chung-Kuan Cheng, and Ronald Graham. Constructing zero-deficiency parallel prefix circuits of minimum depth. *ACM Transactions on Design Automation of Electronic Systems*, V(to appear), accepted in 2005.

[29] R. Zimmermann and D.Q. Tran. Optimized synthesis of sum-of-products. In *Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 867–872. IEEE, 2003.

[30] Reto Zimmermann. Non-heuristic optimization and synthesis of parallel-prefix adders. In *Proc. Int. Workshop on Logic and Architecture Synthesis (IWLAS'96)*, 1996.