A Family of Adders

Simon Knowles
Element 14, Aztec Centre, Bristol, UK
sknowles@e-14.com

Abstract

Binary carry-propagating addition can be efficiently expressed as a prefix computation. Several examples of adders based on such a formulation have been published, and efficient implementations are numerous. Chief among the known constructions are those of Kogge & Stone and Ladner & Fischer. In this work we show that these are end cases of a large family of addition structures, all of which share the attractive property of minimum logical depth. The intermediate structures allow trade-offs between the amount of internal wiring and the fanout of intermediate nodes, and can thus usually achieve a more attractive combination of speed and area/power cost than either of the known end-cases. Rules for the construction of such adders are given, as are examples of realistic 32b designs implemented in an industrial 0u25 CMOS process.

1. Introduction

There are many ways of formulating the process of binary addition. Each different way provides different insight and thus suggests different implementations. Examples are Weinberger & Smith's carry-lookahead adder [Wein58], Nadler's pyramid adder [Nadl56], Sklansky's conditional sum adder [Skla60], Bedrij's carry-select adder [Bedr62], and Ladner & Fischer's prefix adder [Ladn80]. For a general introduction see [Omon94]. The prefix formulation is particularly attractive because it is easily expressed and suggests very efficient implementations, ie. adders based on this formulation can be attractively fast and compact when implemented in VLSI.

This paper is organised as follows. The next section reprises the prefix formulation of addition, and introduces the key properties of associativity and idempotency which make this formulation so flexible. Section 3 covers existing variants of the prefix addition algorithm and their corresponding implementations. Then in Section 4 we introduce a new family of addition structures, all of which have minimum logical depth like Ladner & Fischer's adder, but which express different trade-offs between area and speed. The Kogge-Stone [Kogg73] and Ladner-Fischer [Ladn80] adders are the end-cases of this family. Section 5 tabulates experimental data showing the range of performances available from the new family of adders when implemented in a modern CMOS technology. Finally Section 6 introduces some less-regular structures which might be advantageous in specific circumstances.

2. Addition as a Prefix Problem

We wish to compute a sum S=A+B. We will use capital letters to represent binary words, small letters to represent bits, and subscripts to indicate arithmetic weight, increasing from 0 at the lsb. Thus c_i signifies a carry into bit i, and $a_{4..0}$ signifies the 5 lsb's of A. Operands A and B have n bits, so sum S has n+1 bits. For simplicity we assume there is no input carry c_0; this can be easily accommodated and does not affect the thesis of the paper.

For n-bit addition, n a power of 2, a minimum-depth prefix adder comprises $1+\log_2 n$ unate logical stages, plus 1 non-unate logical stage. In CMOS technology this maps to $3+\log_2 n$ inverting gate stages, plus whatever buffering stages are warranted by the design criteria. The first and last logical stages are basically the same for all types; prefix adders are distinguished by the structure of their intermediate stages. We will describe the Ladner-Fischer structure initially, which is in some sense a basis for the others.

The first stage of the adder computes carry generate (g), propagate (p), and kill (k) terms for each bit according to the relations:

$$g_i = a_i \cdot b_i$$
$$k_i = \overline{a_i + b_i}$$
$$p_i = a_i \oplus b_i$$

The g and k terms are then combined in $\log_2 n$ logical stages (to be described shortly) to produce carries (c_i) into each bit position using the iterative relation:

$$c_{i+1} = g_i + \bar{k}_i c_i$$

The final stage computes sum bits (s_i) as:

$$s_i = p_i \oplus c_i$$

The first and last stages are intrinsically fast because they involve only simple operations on signals local to each bit position. The intermediate stages embody the long-distance propagation of carries, so the performance of the adder hinges on this part. In a prefix adder this part is constructed of nodes which perform the prefix operation '•':

$$\left(\frac{g}{k}\right)_i \bullet \left(\frac{g}{k}\right)_j = \left(\begin{array}{c} g_i + \bar{k}_i . g_j \\ \hline \bar{k}_i . \bar{k}_j \end{array}\right)$$

In logical terms, the prefix operator consists of an AND gate and an AND-OR gate.

The carry into any bit position can be computed as a chain of prefix operations:

$$\left(\frac{c_{i+1}}{\bar{k}_i . \bar{k}_{i-1} . \bar{k}_{i-2} \ldots \bar{k}_0}\right) = \left(\frac{g}{k}\right)_i \bullet \left(\frac{g}{k}\right)_{i-1} \bullet \left(\frac{g}{k}\right)_{i-2} \ldots \bullet \left(\frac{g}{k}\right)_0$$

Naively this can be implemented as a ripple-carry process, but the prefix operator '•' has two essential properties which allow greater parallelism, and hence faster circuits. Firstly it is associative:

$$\left(\frac{g}{k}\right)_{h \ldots j} \bullet \left(\frac{g}{k}\right)_{j \ldots k} = \left(\frac{g}{k}\right)_{h \ldots i} \bullet \left(\frac{g}{k}\right)_{i \ldots k}$$

where h>i≥j>k. Secondly it is idempotent, so:

$$\left(\frac{g}{k}\right)_{h \ldots j} \bullet \left(\frac{g}{k}\right)_{i \ldots k} = \left(\frac{g}{k}\right)_{h \ldots k}$$

Associativity allows the precomputation of sub-terms of the prefix equations, which means the serial iteration implied by the prefix equation above can be parallelized. Idempotency allows these sub-terms to overlap, which provides some useful flexibility in the parallelization.

The Ladner-Fischer scheme exploits the associativity property (but not the idempotency property) by constructing a binary tree of prefix operators. The structure is succinctly represented by the prefix graph of Figure 1, which shows the lateral connectivity required between nodes at each stage of a 32b adder. The inputs are at the top, outputs at the bottom, lsb at the right hand side. The graph only shows the lateral connections; there are also implicit vertical connections between nodes in the same column. The first row of nodes computes the *g, p, k* terms. In subsequent rows, the nodes having lateral connections implement the prefix operator, while those with no lateral connections are just place-holders. The final row of nodes also computes the sum bits. The final carry-out (the msb of the sum) emerges from the most significant node at the last level but is not represented explicitly in the graph. This is clearer in Figure 2, which shows a detailed gate circuit for a 32b adder corresponding to the graph of Figure 1. Note that each lateral connection in the graph corresponds to 2 wires (*g, k*) in the circuit, except at the last level where the final *k* term is not required.

Adders in which the computation of carries is based on the prefix equations above are naturally called 'prefix adders'. Those which compute multiple sub-terms in parallel by exploiting the associativity property are called 'parallel prefix adders'. Many of these also exploit the idempotency property, the most obvious example being the Kogge-Stone adder introduced in the next Section.

3. The Evolution of Parallel Prefix Adders

Ladner and Fischer [Ladn80] introduced the minimum-depth prefix graph described in the previous Section, based on earlier theoretical work by Ofman [Ofma63]. The longest lateral fanning wires go from a node to n/2 other nodes. Capacitive fanout loads become particularly large for later levels in the graph as increasing logical fanout combines with increasing span of the wires. In CMOS implementations such as Figure 2 buffering inverters are added appropriately to support these large loads. There is a corresponding cost in delay.

Kogge and Stone [Kogg73] addressed this fanout issue by introducing the 'recursive doubling' algorithm, which leads, for example, to the 32b prefix graph of figure 3. The Kogge-Stone scheme uses the idempotency property to limit the lateral logical fanout at each node to unity, but at the cost of a dramatic increase in the number of lateral wires at each level. This is because there is massive overlap between the prefix sub-terms being pre-computed. The span of the lateral wires remains the same as for the Ladner-Fischer structure, so some buffering is still usually required to accommodate the wiring capacitance, even though the logical fanout has been minimized.

Other researchers have sought to address the problem of high fanout nodes in the Ladner-Fischer structure by allowing the logical depth of the structure to increase. Brent & Kung [Bren82] proposed the explicit provision of a set of binary fanout trees such that the lateral fanout of each node is restricted to unity, as for the Kogge-Stone graph, but without the explosion of wires. Although attractive at an abstract level this approach makes little sense in a practical CMOS context. Firstly the unit logical fanout limitation is arbitrary and somewhat extreme. Secondly, and of course inseparably, the construction makes no allowance for the fact that much of the capacitive load is due to the span of the wires rather than the number of driven nodes. Practically, it is more efficient to insert buffers as required into the Ladner-Fischer adder, in the manner of Figure 2, than to use the Brent-Kung scheme.

Han and Carlson [Han87] give a good overview of prefix addition formulations, and present their own hybrid synthesis of the Ladner-Fischer and Kogge-Stone graphs. Again this trades an increase in logical depth for a reduction in fanout. It is effectively a higher-radix variant of the Kogge-Stone scheme. Kowalczuk, Tudor & Mlynek [Kowa91] achieve a similar compromise by serializing the prefix computation occurring at the higher fanout nodes, and Beaumont-Smith & Burgess [Beau97] combine this idea with the Han-Carlson scheme.

All these latter papers allow the logical depth, and hence the delay, of the adder to increase in exchange for reductions in fanout or wire flux. This paper illustrates that these gains are available without increasing logical depth from the minimum used in the Ladner-Fischer and Kogge-Stone structures. Lynch and Swartzlander [Lync91] present a minimum-depth prefix addition algorithm which exploits the idempotency of the prefix operation to achieve efficient variants of the Ladner-Fisher formulation for non-power-of-2 operand width. The adders presented in this paper all have power-of-2 operand width, and could likewise be extended to non-power-of-2 widths by the Lynch-Swartzlander scheme.

4. A New Family of Adders

The main purpose of this paper is to introduce Figure 4, which shows a number of new minimum-depth prefix graphs for addition. Graphs are shown for 4b, 8b, and 16b adders. The sets are bounded at either end by the Ladner-Fischer and Kogge-Stone graphs. The graphs are uniquely labelled by listing the lateral fanouts at each level, from the stage nearest the output, back towards the input. Thus the Ladner-Fischer graphs are labelled [2,1], [4,2,1], and [8,4,2,1] for 4b, 8b, and 16b adders respectively, and the corresponding Kogge-Stone graphs are labelled [1,1], [1,1,1], and [1,1,1,1].

The construction of other members of the family is fairly intuitive, with smaller graphs being combined to form progressively larger ones, and the idempotency property being used to fill any gaps in the prefix computation. The following rules govern the construction of these graphs: let the wiring levels between rows of graph nodes be numbered from j=0 at input to $j=\log_2(n)-1$ at output, then:

- Lateral wires at the j^{th} level span 2^j bits.
- The lateral fanout at the j^{th} level is a power of 2 between 1 and 2^j inclusive.
- The lateral fanout at the j^{th} level cannot exceed that at the $(j+1)^{th}$ level.

These rules are sufficient to determine the number of possible minimum-depth graphs of this type (see the next section for further possibilities which are not strictly 'of this type'). For power-of-2 operand widths the number of graphs is as follows:

operand width (bits)	number of basic minimum-depth graphs
4	2
8	5
16	14
32	42
64	132
128	429
256	1430

At 4b width, the Ladner-Fischer and Kogge-Stone graphs are the only ones having minimum depth. Beyond 4b several new possibilities emerge which offer trade-offs between fanout and number of wires at each level. Figure 4 shows all such graphs for 4, 8, and 16 bit operands. Figure 5 shows an example graph for a new 32b adder having structure [4,4,2,2,1], and Figure 6 shows a gate circuit implementing such an adder.

5. Practical Results

Several of these 32b adder designs have been taken through an industrial structured-custom design flow to layout in a 0µ25 6-metal CMOS process with 1µm contacted wire pitch. The selection includes the Ladner-Fischer [16,8,4,2,1] and Kogge-Stone [1,1,1,1,1] extrema, and four intermediate forms. Speed and area results are tabulated below.

The number of buffering inverters added at each level to optimize speed is given in a way which corresponds to the labelling of the graphs. These inverters can be seen clearly in Figures 2 and 6. For the Kogge-Stone [1,1,1,1,1] and Figure 6 [4,4,2,2,1] examples, two different buffering schemes are examined.

	Structure	Buffering	Delay (ref invs)	Length (μm)	Transverse wire flux	
					By level	Total
Ladner-Fischer (fig 2)	[16,8,4,2,1]	[2,1,1,0,0]	13.7	38	[1,2,2,2,2]	9
-	[16,4,2,2,1]	[2,1,1,0,0]	13.2	38	[1,4,4,2,2]	13
-	[16,2,2,2,1]	[2,1,1,0,0]	13.0	41	[1,8,4,2,2]	17
(fig 6)	[4,4,2,2,1]	[1,1,0,0,0]	13.2	35	[4,4,4,2,2]	16
-	[4,4,2,2,1]	[1,1,1,0,0]	12.7	39	[4,4,4,2,2]	16
-	[2,2,2,1,1]	[1,1,1,0,0]	12.1	46	[8,8,4,4,2]	26
Kogge-Stone	[1,1,1,1,1]	[1,1,0,0,0]	12.1	63	[16,16,8,4,2]	42
	[1,1,1,1,1]	[1,1,1,0,0]	11.8	63	[16,16,8,4,2]	42

To remove process and environmental dependencies, critical path delays are normalized to the average of the rise and fall delays of a reference inverter under the same conditions. The reference inverter delay is measured in a tree of identical inverters such that each inverter drives 4 others with zero wiring load. For a contemporary 0μ25 CMOS process at worst-case design conditions (worst-case process params, 125C, 2V0) this reference inverter delay will typically be in the range 120-150ps; in favourable conditions (typical process params, 25C, 2V5) it will be 40-50% lower. As expected, the Kogge-Stone adder is the fastest and the Ladner-Fischer the slowest, although the difference between them is less than 15%.

The adders are laid out as datapath slices using 10μm high cells at 2 cell rows per bit. All examples are therefore 640μm high. Those nearer the Kogge-Stone extreme are wire-limited, with both Kogge-Stone examples requiring 80% more area than the smallest here ([4,4,2,2,1] buffered [1,1,0,0,0]). Because of the differences in buffering requirements, the Ladner-Fischer adder turns out not to be the smallest, by a small margin.

6. Hybrid Schemes

The adders shown in Figure 4 are somewhat homogeneous in construction, in that the fanout is the same for all fanning nodes within a particular level. This is not absolutely necessary, provided the construction rules set out in the Section 4 are respected. Other, less regular, structures exist - for example the 16b graph of Figure 7. Here the critical path of an adder close to the Kogge-Stone extreme (in fact [2,1,1,1]) has been preserved, while paths which are intrinsically shorter and therefore less critical have been pared back towards the Ladner-Fischer extreme.

In a structured layout environment the maximum (rather than average) wiring flux at each level will tend to determine the layout density. Therefore it is unlikely that irregular hybrids such as Figure 7 would offer an area advantage in comparison to the regular version having the same structure on its critical path (in this case structure [2,1,1,1] from Figure 4). The total amount of wiring has nevertheless been reduced so a power advantage might accrue. In the random placement and routing environment typical of contemporary ASIC design flows there might be some area advantage, but in that context the robustness of the other comparisons made in this paper is reduced.

7. Conclusions

The trade-off between circuit speed and area in CMOS is frequently somewhat sharp, and designers of high performance chips often pay a high price in area to close the last few percent on their speed targets. Given an efficient baseline implementation of any logic circuit, this final speed-up is usually achieved by the pervasive introduction of parallel, logically-redundant paths. This is what we are doing in moving from a Ladner-Fischer to a Kogge-Stone formulation for prefix addition; the introduction of redundancy is reflected in the reliance on the idempotency of the prefix operation. The results presented here indicate that prefix adders implemented in contemporary CMOS processes from static gates can achieve a real speed-up of about 15% by this means, from the Ladner-Fischer baseline. But the cost of achieving this full potential by adopting the Kogge-Stone structure is quite large in terms of wiring, and hence area and power. This paper has introduced a family of adders which span a range of speed *vs* area/power trade-offs between these two extremes. All these adders have minimum logical depth. Circuits quite close in speed to the Kogge-Stone adder are available at significantly lower wiring cost.

References

[Know98] S.C. Knowles; 'Designing Addition Circuits' British Patent Application, 17 Aug. 98.

[Wein58] A. Weinberger, J.L Smith; 'A Logic for High-Speed Addition' Nat. Bur. Stand. Circ., 591:3-12, 1958.

[Nadl56] M. Nadler; 'A High-Speed Electronic Arithmetic Unit for Automatic Computing Machines' Alta Technica (Prague), 6:464-478, 1956.

[Skla60] J. Sklansky; 'Conditional-Sum Addition Logic' IRE Trans., EC-9:226-231, June 60.

[Bedr62] O.J. Bedrij; 'Carry-Select Adder' IRE Trans., EC-11:340-346, June 62.

[Omon94] A.R. Omondi; 'Computer Arithmetic Systems' ISBN 0-13-334301-4, 1994.

[Ladn80] R.E. Ladner, M.J. Fischer; 'Parallel Prefix Computation' JACM, 27(4):831-838, Oct. 80.

[Kogg73] P.M. Kogge, H.S. Stone; 'A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations' IEEE Trans., C-22(8):786-793, Aug. 73.

[Ofma73] Y. Ofman; 'On the Algorithmic Complexity of Discrete Functions' Soviet Physics – Doklady, 7(7):589-591, Jan 63.

[Bren82] R.P. Brent, H.T. Kung; 'A Regular Layout for Parallel Adders' IEEE Trans., C-31(3):260-264, March 82.

[Han87] T. Han, D.A. Carlson; 'Fast Area-Efficient VLSI Adders' 8th IEEE Symp. Computer Arithmetic, Como Italy, pp. 49-56, May 87.

[Kowa91] J. Kowalczuk, S. Tudor, D. Mlynek; 'A New Architecture for Automatic Generation of Fast Pipelined Adders' ESSCIRC, Milano Italy, pp. 101-104, Sept 91.

[Beau97] A. Beaumont-Smith, N. Burgess; 'A GaAs 32-bit Adder' 13th Symp. Computer Arithmetic, Asilomar California, pp. 10-17, June 97.

[Lync91] T. Lynch, E.E. Swartzlander Jr; 'The Redundant Cell Adder' 10th IEEE Symp. Computer Arithmetic, Grenoble France, pp. 165-170, June 91.

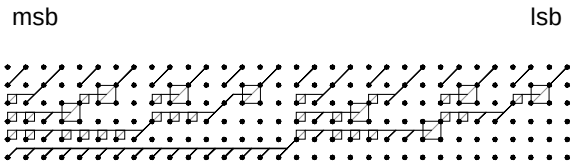

Figure 1: 32b Ladner-Fischer graph [16,8,4,2,1]

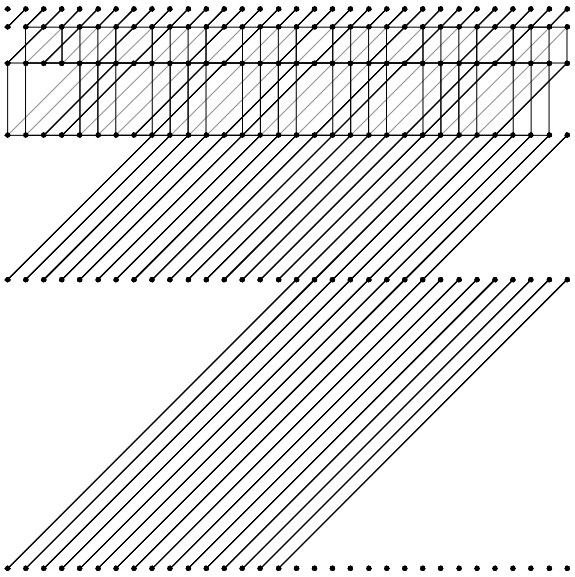

Figure 3: 32b Kogge-Stone graph [1,1,1,1,1]

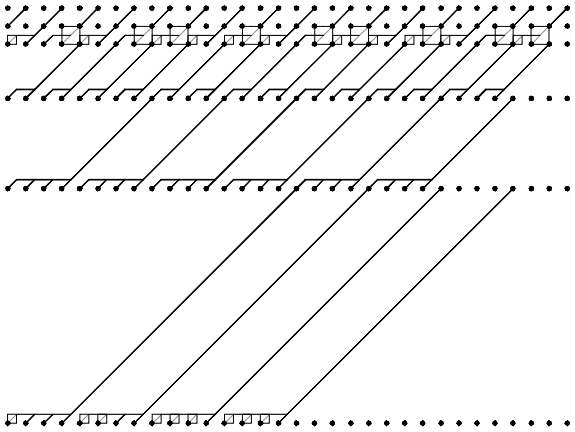

Figure 5: 32b graph [4,4,2,2,1]

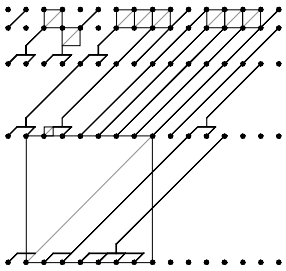

Figure 7: 16b hybrid graph

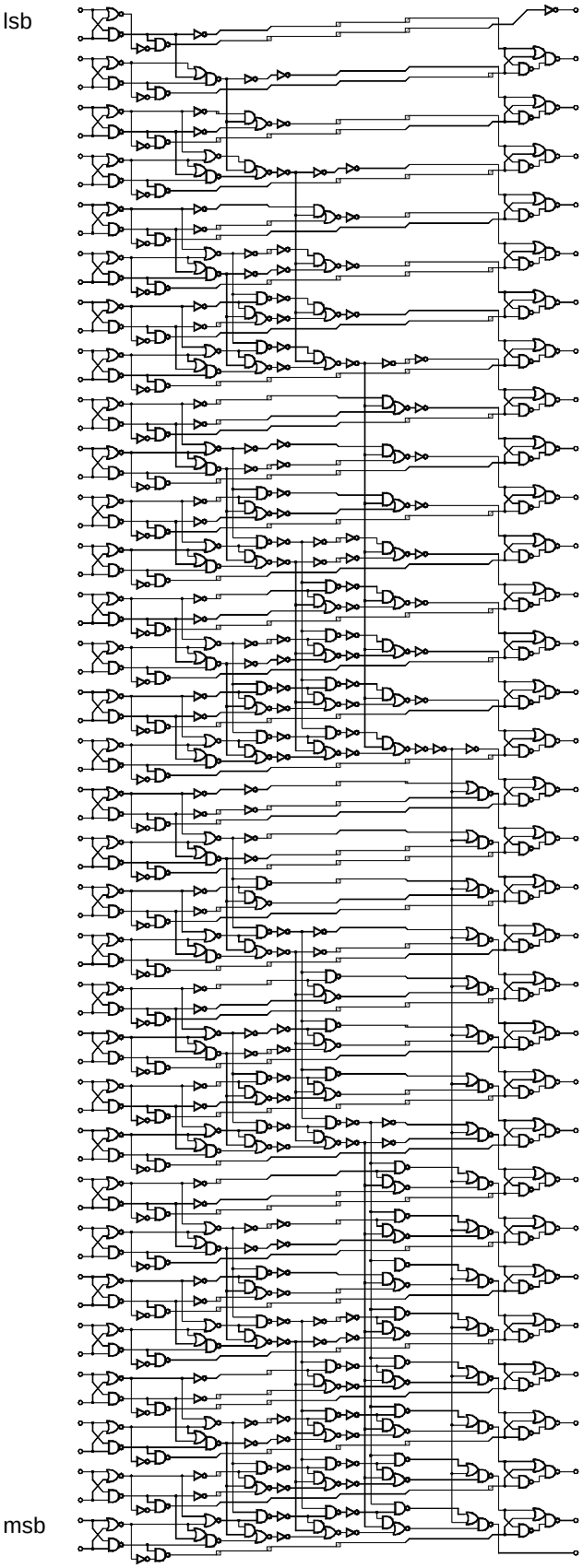

Figure 2:
Ladner-Fischer
adder [16,8,4,2,1]

Figure 4:
4b, 8b, and 16b
minimum-depth graphs

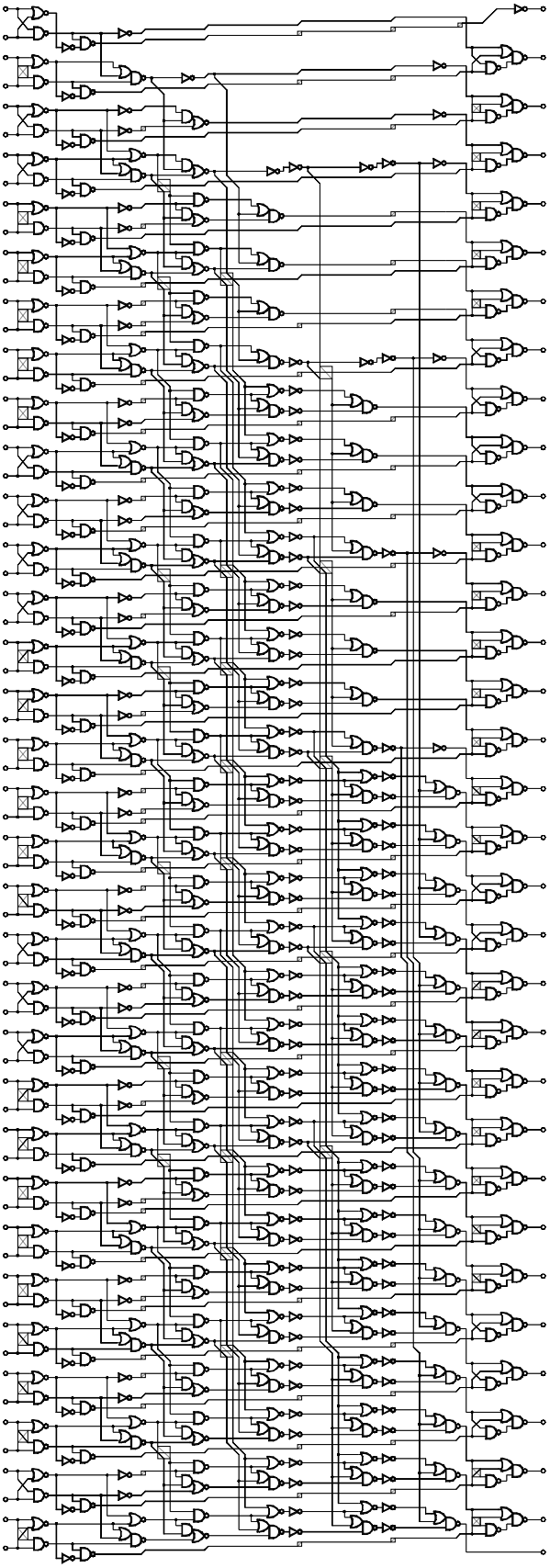

Figure 6:
Adder [4,4,2,2,1]