A Multiplexer-Based Digital Passive Linear Counter (PLINCO)

Skyler Weaver, Benjamin Hershberg, Pavan Kumar Hanumolu, and Un-Ku Moon

School of EECS, Oregon State University, 1148 Kelley Engineering Center, Corvallis, OR 97331. USA.

phone: 1-541-908-1731

email: skyler@skylerweaver.net

Abstract:

A ones adder is an important circuit block that is required in many varying applications. This work proposes a design that largely relies on passive transmission-gate multiplexers. Many variations are suggested that can inherently generate a thermometer coded output or one-hot encoded output. The proposed structure has area and power that increases with order n^2 for a n number of inputs. A folding technique is then suggested that reduces the area/power to order $n \log(n)$. The folded PLINCO also has a cell-based structure that aids in layout and makes it possible to be added to a digital standard cell library.

Keywords: low power; digital circuits; ones adder; thermometer code

I. Introduction

This paper introduces a new architecture for implementing a digital ones adder and similar circuits. The original motivation of this work was with regard to bubble correction in flash analog-to-digital converters (ADCs) and therefore will be discussed in this context; however, the applications of a new structure for digital addition are much broader than the specific context considered here.

Flash architecture ADCs are the preferred ADC architecture for high-speed applications [1, 2]. Flash-based ADCs consist of an input that is connected to a set of comparators that are all given different references, which are equally spaced by a least-significant-bit (LSB) voltage. This can be achieved by using an equal-valued resistor string. Each clock cycle, every comparator produces either a "high" or "low" result (i.e. 1 or 0) depending on whether the input is higher or lower than the reference given to that comparator. This implies that there are many (2^N–1) (where N is the number of bits) comparator outputs; however, if there is negligible comparator offset and the references are monotonic, then there will be only one transition between 1 and 0 outputs, e.g. "1111100" as a possible 3-bit flash output. This is called a thermometer code. The 1-to-0 transition can be found, for example, by adding digital XOR gates with one input connected to a comparator output and the other input connected to the adjacent comparator output. All XOR outputs will be 0 except for the XOR gate with differing inputs, e.g. "0000100". This resulting code of all 0s and a single 1 is called a "one-hot code," since only one wire has a digital 1. A one-hot coded result is typically converted to binary using a lookup-table ROM or a wired-OR matrix [3].

If comparator offset is large enough to not be negligible (or the monotonicity of the references is not guaranteed), then there can be "bubble errors" in this transition. First-order bubble errors (those occurring adjacent to the 1-to-0 transition, e.g. "1110100") can be easily corrected for, but if higher-order bubble errors can occur, the digital bubble error correction becomes increasingly complex. At some point it is necessary to sum the total number of 1 outputs with a ones adder to avoid catastrophic glitches from the decoder [3]. Since there is a steady trend in integrated circuits to be scaled in their dimensions [4, 5], we can expect scaled transistor variability to increase at each new technology node [6]. This device variation leads to large random offsets in comparators; so large that it

has been proposed to use random comparator offsets *as* comparator references (instead of a resistor string) [7–9]. Since in these designs there is no thermometer coded output, conventional digital adder trees [1, 10] must be used as ones adders. The power and area required for the ones adder is significant in this type of design and can even exceed the power used by the comparators themselves. In this paper we present an alternative to a full-adder-based ones adder by making a multiplexer-based structure that performs the same function.

II. Thermometer code generating PLINCO

Combining multiplexers (muxes) as shown in Fig. 1 will create a circuit we are calling a *p*assive *lin*ear *co*unter (PLINCO). "Linear counter" since the output is a count of the number 1s in the input, and the output is not inherently binary. In order to make this circuit passive (and more efficient), each digital mux is implemented as a passive transmission gate Fig. 2. The digital input to the PLINCO examples in this paper can be thought of as the output of a 3-bit flash ADC with a very nonmonotonic output (here the flash output is completely scrambled), or as some random digital vector that needs to be summed or sorted. Each digital input is a select line for one row of muxes (denoted as S and S-bar in Fig. 2). The unary-weighted digital word that appears above a row of muxes is then shifted right or shifted left depending on whether the select line is 0 or 1, respectively. If shifted left, a digital 1 is shifted in from the right; if shifted right, a digital 0 is shifted from the left. In this manner, after each row of muxes the digital word grows by one bit until the final output is the same length as the original PLINCO input; however, the output is now sorted into a perfect thermometer code.

A thermometer code generating PLINCO can also be used in the case where there are two thermometer codes coming from separate ADCs that must be combined into a single thermometer code. With out using a PLINCO the thermometer codes must be converted to binary, summed with binary addition, then converted back into a thermometer code. This technique of using a multiplexer-based approach was used in [11] to combine the outputs of two quantizers whose combined output needed to be fed to a dynamic element matching block in the form of a thermometer code. This approach was found to be faster and more efficient than a thermometer-to-binary-to-thermometer approach.

III. Majority-vote PLINCO

In some instances it may only be important to know whether there are more 1s or more 0s, rather than the total number of 1s; we call this a majority-vote [12]. A majority-vote circuit can be easily implemented as a thermometer code generating PLINCO where only the middle of the thermometer code output is important, therefore the superfluous muxes can be removed. An example of a 7-input majority-vote PLINCO is shown in Fig. 3.

IV. One-hot code generating PLINCO

As indicated in the introduction, sometimes a thermometer code is less desirable than a one-hot code since the latter can connect directly into a lookup-table or wired-OR matrix. By flipping each mux in a thermometer code generating PLINCO upside-down, the result is a PLINCO that gives a one-hot coded output (Fig. 4). In this variation, the digital word a single digital 1 that is shifted left or right depending on the next row's select line. The final location of the single 1 in the digital output indicates the total number of 1s present in the input to the PLINCO.

This same function could be implemented as a shift register with a single bit as digital 1 and the rest as digital 0. One at a time, each digital input value shifts the register left or right. Once all of the digital input values have been processed, the location of the 1 in the shift register tells you the total number of left shifts minus the total number of right shifts. This shift register implementation sequentially processes the input vector, whereas PLINCO spatially processes the input vector.

One major advantage of the one-hot code generating PLINCO over the thermometer code generating PLINCO is that since each mux either propagates the single 1 or merely propagates nothing. Since a NMOS transistor will pass a digital 0 quickly but a PMOS transistor will pass a digital 1 quickly, the muxes do not need both NMOS and PMOS transmission paths if only a single digital polarity is propagated. In fact, all of the muxes can be replaced with the simpler version shown in Fig. 5 (assuming that the outputs are reset to 0 each cycle).

V. Folded PLINCO

The two previous PLINCO architectures are very efficient for a small number of inputs, yet if the number of inputs n is very large, the number of muxes required increases with quadratically, proportional to n^2. This implies that area and power will follow the same order. Also, since PLINCO is defined as being passive, each row of muxes increases the time-constant such that buffers would need to be placed every certain number (e.g. 8) of mux rows. To combat both the quadratic increase in area and the need for buffers, a PLINCO can be folded into cells of many smaller PLINCOs.

Fig. 6 visualizes this motivation and the wasted area for a large number of inputs. Fig. 6(a) shows a one-hot generating PLINCO with some large number of inputs. Once the single digital 1 has propagated sufficiently through the PLINCO the number of muxes required to continue propagating the digital value is less (in some cases much less) than the number of muxes that actually remain. Fig. 6(b) is a folded PLINCO with the same number of inputs. The number of muxes are reduced by partitioning the PLINCO at every boundary where the number of mux outputs reaches some value. Combination logic can decide whether the propagating digital value is in the most-left portion or the most-right portion of the mux outputs. The knowledge of most-left vs. most-right is akin to the carry bit from a full adder and can be processed elsewhere, such as in another PLINCO. This same logic will correctly propagate the digital value to the next, smaller row of muxes, and the propagation continues. By logically propagating to the next row, the folding logic also acts as a buffer to the next stage, which would be required anyway.

The other advantage is that once it has been decided to set the partitioning such that each PLINCO stage propagates to some number of outputs, then folds the output by some factor, the overall design can be constructed by cascading multiple folded PLINCO "cells." An example of two 72-input folded PLINCO designs with different partitioning parameters can be seen in Fig. 7. The first design (Fig. 7(a)) allows each stage to propagate the output to 16 possible mux outputs. Combinational logic will then decide if the signal is in the leftmost 8 outputs or the rightmost 8 outputs. This decision is then passed to another PLINCO; however, the "carry" has a bit-weight of 8. Another design (Fig. 7(b)) allows each stage to propagate the output to only 8 possible mux outputs.

Therefore the carry bit from this folding logic is only worth a bit-weight of 4. It can be seen that in this design, the PLINCO that processes the carry information also contains folding logic that is again passed to an additional PLINCO. Area and power for a folded PLINCO is proportional to $n \cdot \log(n)$, instead of n^2, on the number of inputs, making it more viable when the number of inputs is very large. The optimum number of inputs per cell and folding factor will depend on a variety of factors. Folding saves area by eliminating muxes; however, the folding logic will have some area and power overhead. The ratio of this overhead to the overall savings will drive the choice of inputs per cell and folding factor to achieve an optimized design.

An implementation of a 4-input folded PLINCO cell with a folding factor of 2 can be seen in Fig. 8. Here it can be seen that the combinational logic passes on either the leftmost 4 PLINCO outputs or the rightmost 4 outputs, yet their relative position is maintained. The reason that there are two 4-input NOR gates to generate the carry bit is to generate both the "S" and "Sbar" signals required for the muxes. A transistor-level implementation of the same PLINCO cell can be seen in Fig. 9. The digital gates are implemented as dynamic logic gates to save area, and dynamically reset the PLINCO outputs to digital 1, since the muxes will only pass a digital 0. In this design a digital 0 was chosen to be the polarity that is propagated since it allows the use of NMOS only muxes which should be faster is a typical CMOS technology.

When making a comparison to a full-adder based design, it is interesting to note that a full adder is actually a subset of a folded PLINCO design. Consider the implementation of a 2-input folded PLINCO cell with a folding factor of 2 (Fig. 10), and it will be recognized as a conventional full adder. In short, with the invention of the folded PLINCO architecture, a designer has more flexibility in designing a ones adder than being constrained to using only conventional full adders.

To quantitatively compare a folded PLINCO based ones adder to a conventional Wallace adder, a 64-input ones adder was implemented using each architecture. The folded PLINCO cell used was the 4-input cell with a folding factor of 2 seen in Fig. 8. The circuits were implemented in a 0.18μm CMOS technology, and parasitic extraction was obtained for each cell. Simulation of the combinational logic delay and power consumption of the two designed were

measured. As shown in Table I, there is a 37% improvement in power-delay product for the PLINCO design over a conventional Wallace tree. To decrease the delay further it should be possible to include pipelining logic in the folding logic, giving the design a higher throughput at the cost of latency. Depending on the system level requirements, pipelining the PLINCO may be acceptable.

VI. Conclusion

The passive linear counter was presented. If a system has unordered unary-weighted outputs, a ones adder must be used to decode the output. Typically a ones adder is a Wallace tree structure with conventional full adders and introduces significant delay, area, and power consumption. By introducing the PLINCO, a designer has another option to try and find an optimum design. In this paper we have proposed many variations of PLINCOs each with differing benefits. An exciting aspect of the PLINCO is the high design flexibility and the fact that it can be easily modified to be used in many design applications.

Acknowledgment

This work was supported by the Air Force Research Labs (AFRL) and Tektronix, Inc.

References

[1] F. Kaess *et al.*, "New Encoding Scheme For High-Speed Flash ADC's," *Circuits and Systems, IEEE International Symposium on*, pp. 5-8, June 1997

[2] R. J. Van de Plassche, *Integrated analog-to-digital and digital-to-analog converters*, Kluwer Academics Publishers, ch. 3, 1994.

[3] E. Säll, M. Vesterbacka, and K. O. Andersson, "A Study of Digital Decoders in Flash Analog-to-Digital Converters," *Circuits and Systems, IEEE International Symposium on*, pp. 129-132, May 2004.

[4] G. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, 1965, pp. 114-117.

[5] B. Spencer, L. Wilson, and R. Doering, "The Semiconductor Technology Roadmap," *Future Fab Int'l*, vol. 18, Jan. 12, 2005.

[6] T.-C. Chen, "Overcoming Research Challenges for CMOS Scaling: Industry Directions," *Solid-State and Integrated Circuit Tech., Inter. Conf. on*, pp. 4-7, October 2006.

[7] C. Donovan and M. P. Flynn, "A `digital' 6-bit ADC in 0.25 μm CMOS," *Solid-State Circuits, IEEE J. of*, vol. 37, no. 3, pp. 432-437, Mar. 2002.

[8] D. C. Daly and A. P. Chandrakasan, "A 6b 0.2-to-0.9V Highly Digital Flash ADC with Comparator Redundancy," *Solid-State Circuits Conf., IEEE Inter.*, pp. 554-555, 635, February 2008.

[9] S. Weaver *et al.*, "A 6b Stochastic Flash Analog-to-Digital Converter Without Calibration or Reference Ladder," *Solid-State Circuits Conf., IEEE Asian*, November 2008.

[10] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. on Electronics Computers*, pp. 14-17, Feb. 1964.

[11] N. Maghari, S. Weaver, U. Moon, "A +5dBFS Third-Order Extended Dynamic Range Single-Loop $\Delta\sum$ Modulator," *Custom Integrated Circuits Conference (CICC), IEEE proc. of*, pp. 1-4, Sep 2010.

[12] P. Hanumolu, G. Wei, and U. Moon, "A wide tracking range 0.2-4Gbps clock and data recovery circuit," *IEEE Symp. VLSI Circuits*, pp. 88-89, Jun. 2006.

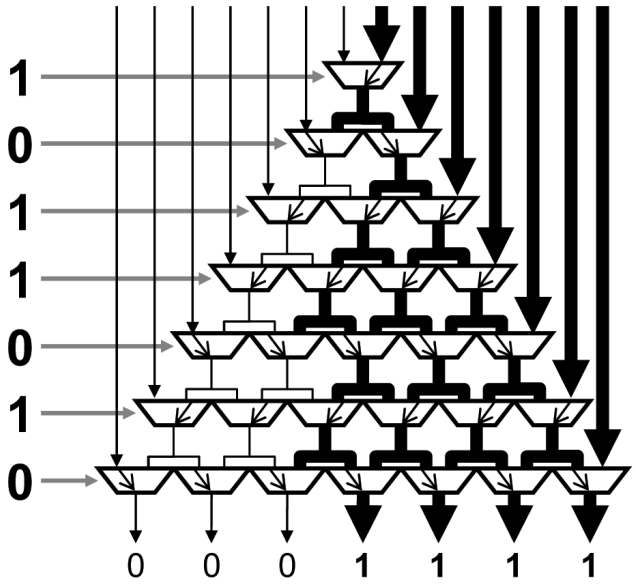


Fig. 1 A thermometer code generating passive linear counter (PLINCO) which has an output with the same number of 1s as the input, but are sorted into a thermometer code

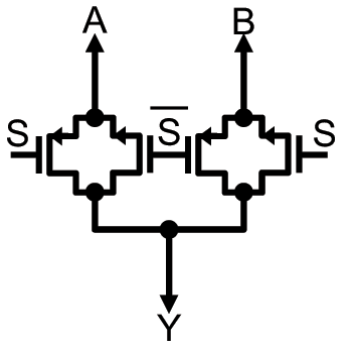


Fig. 2 Two transmission gates form a passive multiplexer

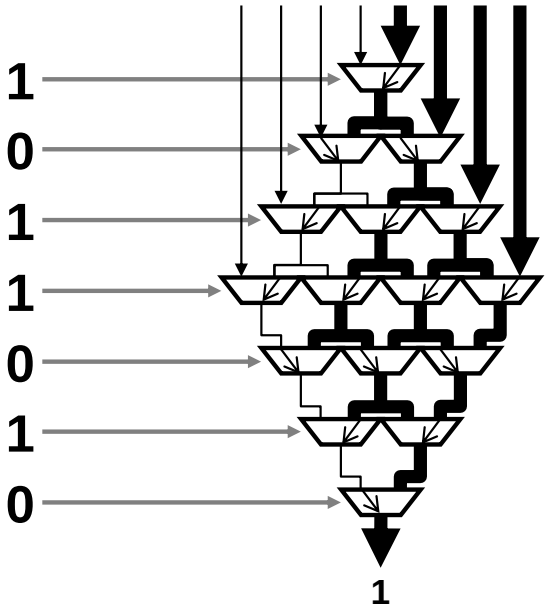


Fig. 3 A majority-vote PLINCO outputs '1' if there were more 1s than 0s in the input

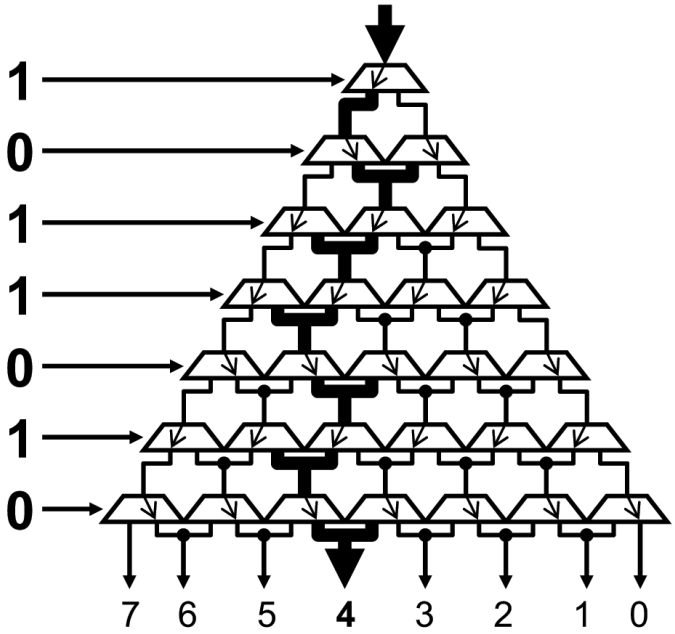


Fig. 4 A one-hot generating PLINCO has one output whose position indicates the number of 1s in the input vector

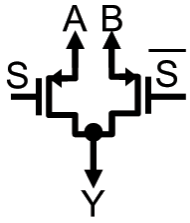
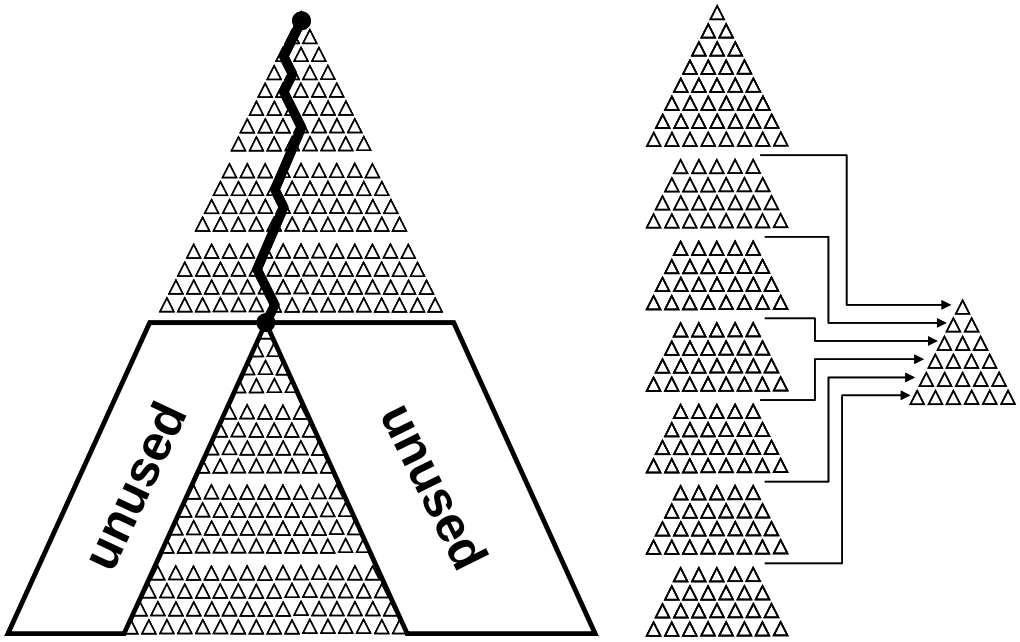
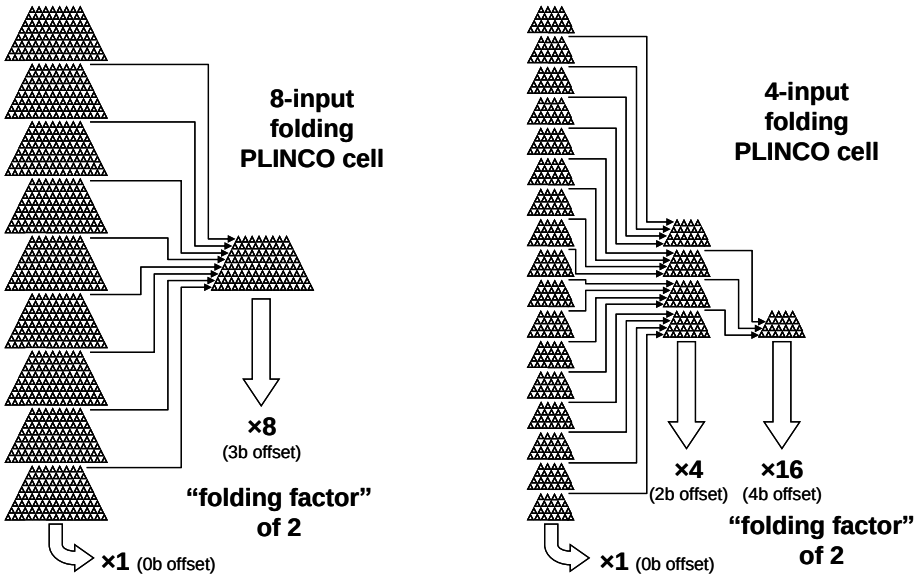


Fig. 5 Two PMOS transistors create a multiplexer that can only propagate a digital 1 quickly, yet has smaller area compared to using transmission gates



(a) (b)

Fig. 6 (a) After an input has rippled through a significant portion of a PLINCO, there are many multiplexers (drawn as triangles) that are not capable of receiving the signal and will be unused (b) By partitioning a PLINCO at various rows and determining the inputs location allows the PLINCO to be folded, dramatically reducing the number of multiplexers required for a large number of inputs



(a) (b)

Fig. 7 (a) A folded PLINCO using 8-input folding PLINCO cells with a folding factor of 2 and 72 digital inputs (b) A folded PLINCO using 4-input folding PLINCO cells with a folding factor of 2 and 72 digital inputs

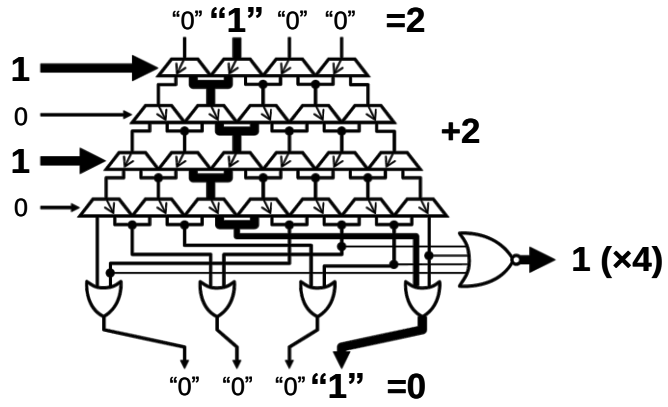


Fig. 8 Gate-level implementation of a 4-input folded PLINCO cell with a folding factor of 2

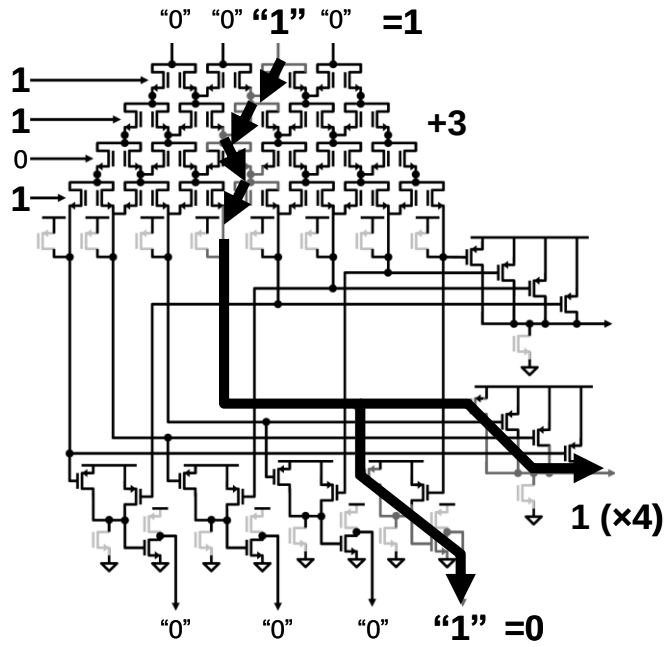


Fig. 9 Transistor-level implementation of a 2-input folded PLINCO cell with a folding factor of 2 using dynamic "domino" logic

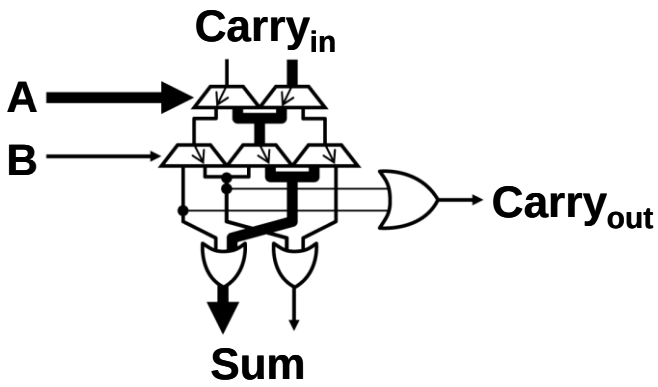


Fig. 10 Gate-level implementation of a 2-input folded PLINCO cell with a folding factor of 2, which is also a PLINCO implementation of a conventional full-adder

	Folded PLINCO	Conventional Wallace Tree
Power	645 μW	1587 μW
Delay	1420 ps	914 ps
Power*Delay	916 fJ	1451 fJ

Table I Power, delay, and power-delay-product (PDP) of two 64-input ones adder in 0.18μm CMOS. The folded PLINCO shows a 37% improvement in PDP.