

# A High Performance, Low Area Overhead Carry Lookahead Adder

James Levy and Jabulani Nyathi

Washington State University

School of Electrical Engineering and Computer Science

P. O. Box 642752, Pullman, WA 99164

## Abstract

*Adders are some of the most critical data path circuits requiring considerable design effort in order to “squeeze” out as much performance gain as possible. Many adder designs manage high performance by reducing the delay of the critical path, an effort that results in high area overhead in most cases. In this paper we present a carry lookahead adder (CLA) with a prediction scheme that results in improved performance and low area overhead. Carry prediction enables for the reduction of the carry circuitry within a block while reducing the delay involved in the generation of the carry-out to the subsequent blocks. We have performed simulations of a 16-bit adder and recorded performance improvements of 67% in propagating the carry and generating the sum when compared with the traditional (fixed group-4) CLA designed in the same technology.*

**Keywords** - carry lookahead, high-performance, adder architecture, carry prediction, area overhead.

## 1. Introduction

High performance data path circuits continue to be a topic of interest more so as technologies are scaled to the nanometer regime. Adders fall under this group and have been the subject of in-depth analysis for decades. Careful optimization of adders and other data path circuits will grow in importance as methods to reduce power dissipation while maintaining or improving performance are sought. Optimization can be performed at the architectural, logic and/or circuit levels. There exist numerous adder implementations each with good attributes and some drawbacks. Examples include ripple carry adders, carry lookahead

adders, carry skip adders, and carry select adders [1] just to name a few and each has numerous variants designed to enhance performance. The advent of wide data paths requiring upwards of 64-bit additions has further intensified the need for adder optimization. As the number of input bits increases so does the delay to propagate the carry from the least significant bit additions to the most significant bit positions to enable generation of the final sum. The desire to reduce the delays associated with carry propagation has resulted in novel adder architectures and implementations [2], [3], [4]. The emergency of new devices such as single electron transistors, resonant tunneling diodes, molecular devices, etc will also see major efforts towards the design of novel high performance data path circuits [5], [6].

In this paper we present two approaches that can be employed to reduce the carry propagation through wide data paths. As proof of concept we base our scheme on a 16-bit carry lookahead adder (CLA) and show that even as the number of input bits increase the carry propagation can still be effectively reduced. We attempt to predict the carry signal for any given inputs at the same time the generate and propagate terms of a carry lookahead adder are computed. It is shown in [1] that the logic terms required for the generation of the carry signals in parallel at every bit position becomes prohibitively large as the number of input bits increases. It is suggested that to reduce the fan-in of the AND gates required to implement these carry-out terms it is best to partition the adder into 4-bit blocks. We have thus taken this approach in our carry prediction scheme.

In Section 2 of this paper we discuss the carry lookahead adder with fixed 4-bit blocks and also examine the structure of the CLA with a prediction scheme. The adder equations are presented along with the equation for the prediction logic. The results of our

study are presented in Section 3 while Section 4 has some concluding remarks.

## 2. Carry Propagation Optimization

In this section we discuss the fixed group-4 propagate and generate carry lookahead adder on which we base our prediction scheme and the new CLA with carry prediction. The logic equations describing the carry out signals generated per bit position show that these signals are produced in parallel [1], however, there is a ripple effect when the sums of the subsequent blocks have to be generated. This is so because the carry-out generated by the 4<sup>th</sup> bit position of a 4-bit block needs to be propagated to become part of the inputs to the next block. The cascading effect of the carry-out leads to considerable delays. The standard carry lookahead adder equations that dictate if the carry will be generated or propagated culminating in the sum being generated are reproduced here just for convenience. We use  $g_i = a_i \bullet b_i$  and  $p_i = a_i \oplus b_i$ , where

$g_i$  and  $p_i$  are respectively generate and propagate signals with  $a_i$  and  $b_i$  being inputs at bit position  $i$ .

The sum is given by:

$$s_i = c_{i-1} \oplus a_i \oplus b_i \quad (1)$$

while the carry of the  $i^{th}$  bit stage is:

$$c_i = g_i + p_i \bullet c_{i-1} \quad (2)$$

Equation 2 when expanded becomes:

$$c_i = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \dots + p_i \dots p_1 c_0 \quad (3)$$

It is common knowledge that the size and fan-in of the gates needed to implement the carry-lookahead scheme increases as the number of input bits grows. Limiting the carry lookahead to four stage blocks

allows for breaking the resultant delay in carry generation. This is the approach taken in this paper.

Expanding equation 2 makes it apparent that each carry bit can be generated independent of the previous carry (equation 3), however, each sum bit depends on the value of the previous carry bit. This observation has led to layered implementations of carry lookahead adders with the first layer producing the  $g_i$  and  $p_i$  terms, the second layer producing the carry bits and the third layer producing the sum. Arranging the adder in this form lends itself well to pipelining, however, as aforementioned an increase in the number of inputs leads to a long delay due to an increase in the number of logic gates required to generate the carry and the dependency of the sum on the carry of the previous bit positions. The fourth carry ( $c_3$ ) for instance will require four AND gates, with one having a fan-in of 4 and one 5-input OR gate. The fifth carry ( $c_4$ ) will have 6 AND gates (one having a fan-in of 6). The description above does not necessarily dictate that the hardware to implement these logic terms must include the number of gates cited. Many innovative ways of breaking this delay and enhancing the performance of adder circuits have been developed and used over the years. Performance has further been improved owing to the capability to scale transistors [7], [8].

### 2.1. Ripple Carry Lookahead

The basic implementation of carry lookahead adders has involved limiting the carry generation circuitry to only produce up to the 4<sup>th</sup> carry bit per block (fixed group-4). This scheme is shown in Figure 1.

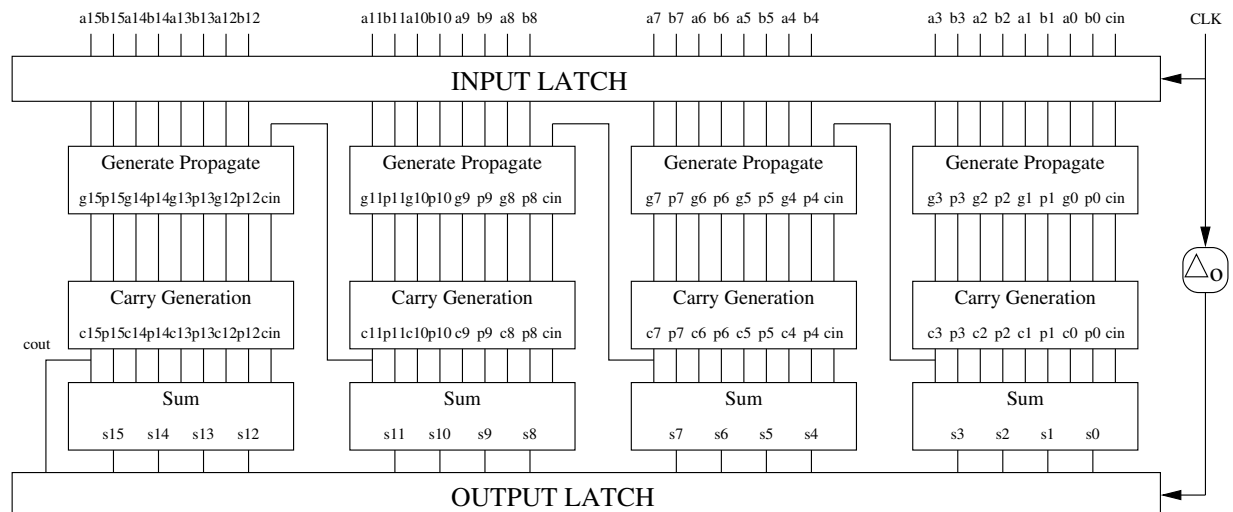


Fig. 1 Three level block diagram of 16-bit CLA without the prediction

Blocks are therefore, cascaded together in a ripple carry adder fashion such that the 4<sup>th</sup> carry bit of each block is the carry input to the succeeding block. Therefore, the 4<sup>th</sup> carry bit of the 1<sup>st</sup> block will be the carry input to the 2<sup>nd</sup> block while the 8<sup>th</sup> carry bit of the 2<sup>nd</sup> block will be the carry input to the 3<sup>rd</sup> block, and so on. This approach results in a ripple effect. The inputs of all the blocks are presented at the same time, enabling each block to commence addition in parallel, however block 1 has to wait for block 0 to produce a carry, similarly block 2 waits on block 1, resulting in a ripple effect as the most significant carry signals of each block must be available as input to the next stage. The focus of many researchers has been to reduce this carry propagation delay in order to improve performance as discussed in [2], [3], [9].

## 2.2. Carry Prediction Scheme

We have designed a carry lookahead adder that attempts to break the ripple effect of the carry signal by predicting the value of the carry-out each block has to propagate to its nearest neighbor. The prediction scheme accepts the inputs at the same layer or gate level as the propagate and generate circuitry and determines if the next block will receive a carry-in of one or zero and provides this carry long before the carry generating block can evaluate this condition. This permits the next block to start adding the received inputs without having to wait for its previous neighbor to produce its most significant carry-out. Our prediction scheme is achieved by considering the upper three bits of both addends at each 4-bit block. Their logic values enable for early detection of what the carry-in of the next block will be. The expression derived to perform this detection is provided in equation 4 and is somewhat similar to the logic used

for propagate and kill signals for the Manchester carry lookahead adder [8]. The representative logic expression for the prediction logic is given by:

$$C_{out} = a_3b_3 + a_2b_2 \bullet (a_3 + b_3) + a_1b_1 \bullet (a_2a_3 + b_2b_3 + a_2b_3 + b_2a_3) \quad (4)$$

This expression obviously has many ANDed terms and requires a maximum fan-in of 4; however, the carry prediction still takes place much earlier than the computed carry at the second layer of the CLA. It is also noted that the advantage is that prediction takes place at the same time propagate and generate signals required to produce the carry are computed. The fan-in of the OR gate also gets large. We show a block diagram of this scheme in Figure 2. The figure shows that the predicted carry is multiplexed with the 1<sup>st</sup> bit carry instead of the 4<sup>th</sup> bit carry. This results from the observation that whenever a prediction cannot be made the resulting 4<sup>th</sup> bit carry-out has the same logic level as the 1<sup>st</sup> carry-out bit. Implementing the prediction scheme in this manner results in the elimination of circuitry required to generate the 4<sup>th</sup> carry. The 1<sup>st</sup> bit carry-out requires less circuitry to generate and can thus be propagated very fast to the next block.

Using the three upper bit pairs result in a situation in which only 8 out of the 64 possible cases cannot predict the carry-out required for the next block. These cases are shown in Table 1. In the event that the carry-out cannot be predicted, the scheme relies on  $p_1$ ,  $p_2$  and  $p_3$  to notify the system that prediction could not be made in which case the next block is made to wait for the computed carry-out, thus the worst case operation is encountered. This worst case delay is better than the one encountered if the approach of Figure 1 is used because the 1<sup>st</sup> carry bit is propagated instead of the 4<sup>th</sup>.

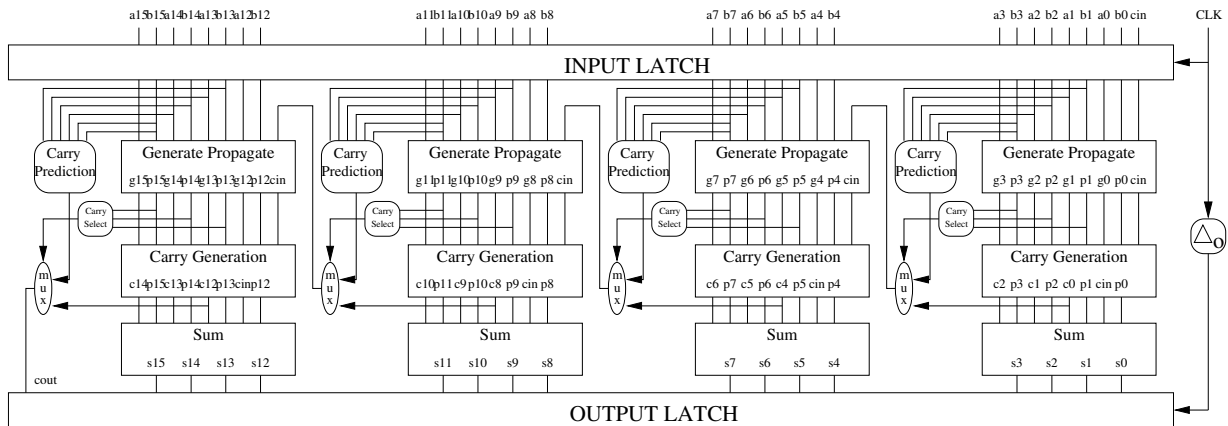


Fig. 2 Three level block diagram of CLA with carry-out prediction based on three upper bits

As stated earlier the 1<sup>st</sup> carry requires less circuitry to generate and is thus available much earlier than the 4<sup>th</sup> carry. We show the performance gain due to this approach in the next section.

**Table 1. Carry-out cannot be predicted with any of these input combination**

a <sub>3</sub>	b <sub>3</sub>	a <sub>2</sub>	b <sub>2</sub>	a <sub>1</sub>	b <sub>1</sub>
0	1	0	1	0	1
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	0	1	0

If the current block (the one whose carry-in could not be predicted) can make a prediction of its most significant carry out, the next block can perform its computations without being affected by its preceding neighbor. This scenario still has some considerable advantage in terms of performance. The worst case operation occurs when no CLA block can make a prediction therefore propagating the carry-out of the adder to the next block. In contrast, in the event that prediction can be made, any CLA block whose carry-input was predicted from the previous block will execute in parallel. Thus, the best case operation is one in which all blocks can predict and the delay is approximately equivalent to that of one 4-bit CLA block. If we consider that the input patterns have an equal probability to occur we can show that only 12.5% of the time prediction would not be achieved and we default to worst case. We thus have 87.5% of operation with the capability to make prediction. This leads to considerable performance improvement.

The prediction circuitry dictates that there be a multiplexer in the design in order to enable for a selection between the predicted carry and the computed carry in the event that prediction cannot be made. The circuitry designed to control the multiplexer is at the level of the carry-out generation circuitry, but can be designed using a single 3-input NAND gate enabling this control signal to be generated much earlier than the carry-out can be computed.

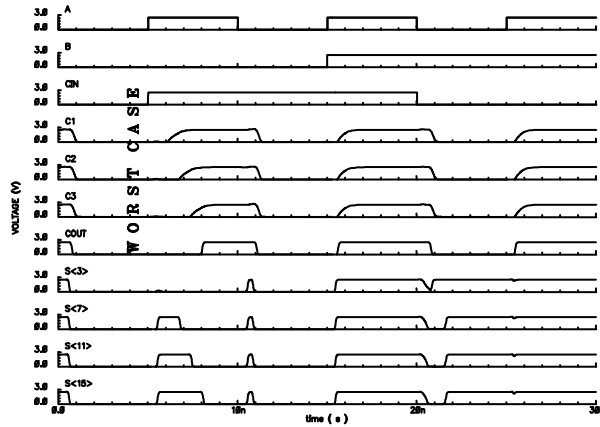
### 3. Results

Simulation results of a 16-bit carry lookahead adder have been performed with the carry prediction scheme

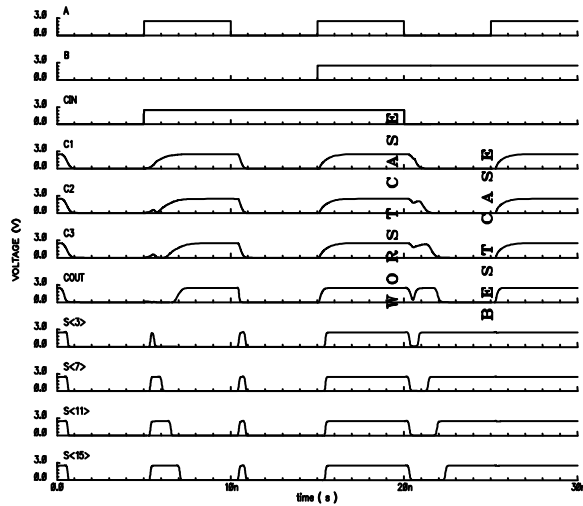
being implemented in pseudo NMOS. Functionality of the scheme was proved mathematically as well as by evaluating simulations. Performance analysis involved evaluating all possible cases that could result in performance degradation. It must be pointed out that performance optimization at transistor level has not been as aggressive as has been reported in [9]. We have not taken advantage of transistor sizing as the focus has been to ensure that the prediction scheme has some benefits. The simulation results were compared to those of a carry lookahead adder without a prediction scheme. Figure 3 shows the delays measured from the adder without the carry-out prediction logic, this delay is for only one transition between blocks. We examine what it takes to propagate the carry signal through the critical path and we show these delays in the figure. After the inputs change it takes 3.083 ns for the sum to be computed. If we consider the CLA with a prediction scheme and propagate the 4<sup>th</sup> carry-out as input to the next block in the event that a prediction cannot be made, the delay becomes longer because of the additional circuitry particularly the multiplexer. The recorded delay is 3.501 ns and this approach is not evaluated any further since the performance gains are negated when prediction cannot be made.

If the CLA with a prediction scheme that considers propagating the 1<sup>st</sup> carry-out in the event that prediction cannot be made is considered, a delay of 2.382 ns is recorded. These are worst case delays when the full 16-bit addition is performed for the three implementations (no prediction, propagating block's 4<sup>th</sup> bit carry-out when prediction fails and propagating the 1<sup>st</sup> bit carry-out when prediction fails). The prediction scheme in which the 1<sup>st</sup> carry bit is forwarded shows a 22% improvement over the case when no prediction is used. The best case of the carry prediction approach yields 67% performance improvement and occurs when prediction is successful at all blocks of the adder. Figures 3 and 4 are traces showing the reported values graphically. To obtain the traces on Figure 4, a case in which the first, third and fourth blocks successfully predict the carry-out signals while the second block fails to predict is shown. Simulations were performed in a 0.25 micron technology.

We have also measured power and determined that our CLA implementation dissipates 1.5% more power than the adder without the prediction scheme. The ability to propagate the 1<sup>st</sup> carry-out to the next block results in reduction of transistor count since the circuitry required to generate the 4<sup>th</sup> carry-out is no longer required. Predicting the carry saves 14% in transistor count, provides performance gains of 22 – 67% and power dissipation increases by only 1.5%.



**Fig. 3 Total delay for CLA without prediction**



**Fig. 4 Delay through a CLA using prediction scheme**

The carry prediction scheme lends itself well to self timed system implementation, variable clock and wave-pipelined clocking methodologies. We consider wave-pipelining the clock [11], [12]. This high performance approach allows the clock to “travel” with the data through all the system’s stages and manages clock skew [13]. The fact that there are cases in which prediction cannot be made during addition implies that we will have variable times in computing the sum. These variations will be accommodated by wave-pipelining the clock since the clock travels with the data. An interested reader is referred to reference [13]. The clocking scheme will increase area, but will not sacrifice performance. In addition this area overhead will not be in the data path. The wave-pipelined clocks are generated based on the select signals and they thus vary according to data path delays.

## 4. Conclusion

In this paper we have presented a carry prediction scheme that enables for the reduction of the carry propagation delay by up to 67% with less area overhead. The prediction scheme allows for the elimination of the circuitry required to generate the 4<sup>th</sup> carry of each block with a 14% reduction in transistor count. The reported performance gains have been achieved without any aggressive transistor sizing. We expect that performance gains would increase with prudent circuit level work. We consider hybrid wave-pipelining for the clock.

## 5. References

- [1] N. West and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective*, 2nd ed., Addison Wesley, NY. 1992, pp. 513-536.
- [2] P. K. Chan, M. D. F. Schlag, C. D. Thomborson, and V. G. Oklobdzija, “Delay Optimization of Carry-Skip Adders and Block Carry-Lookahead Adders,” *10<sup>th</sup> IEEE Proceedings on Computer Arithmetic*, June 26-28, 1991, pp.154–164.
- [3] C-H. Huang, J-S. Wang, C. Yeh and C-J. Fang, “The CMOS Carry-Forward Adders,” *IEEE Journal of Solid-State Circuits*, Vol. 39, NO. 2, February 2004, pp. 327-336.
- [4] Y. Kim and L-S Kim, “64-bit carry-select adder with reduced area,” *Electronics Letters*, Vol. 37, Issue 10, May 10, 2001, pp. 614–615.
- [5] K. Yamamura and Y. Suda, “Novel single-electron logic circuits using charge induced signal transmission (CIST) structures,” *IEEE Transactions on Nanotechnology*, Vol. 2, Issue 1, March 2003, pp. 1–9.
- [6] G. T. Zardalidis, and I Karafyllidis, “Design and simulation of a single-electron full-adder,” *IEE Proceedings on Circuits, Devices and Systems*, Vol. 150, Issue 3, June 6, 2003, pp. 173-177.
- [7] K-H. Cheng; W-S. Lee and Y-C. Huang, “A 1.2V 500 MHz 32-bit Carry-Lookahead Adder,” *8<sup>th</sup> IEEE International Conference on Electronics, Circuits and Systems*, Vol. 2, September 2-5, 2001, pp. 765–768.
- [8] P. Corsonello, S. Perri and G. Cocorullo, “Hybrid carry-select statistical carry lookahead adder,” *Electronics Letters*, Vol. 35, Issue 7 April 1, 1999, pp. 549–551.
- [9] H. Eriksson, P. Larsson-Edefors, and A. Alvandpour, “A 2.8 ns 30  $\mu$ W/MHz area-efficient 32-b Manchester carry-bypass adder,” *IEEE International Symposium on Circuits and Systems*, Vol. 4, May 6-9, 2001, pp. 84–87.

[10] C-J. Fang, C-H. Huang, J-S. Wang and C-W. Yeh "Fast and compact dynamic ripple carry adder design," *IEEE Asia-Pacific Conference on ASIC*, Aug. 6-8, 2002, pp. 25-28.

[11] J. Nyathi and J. G. Delgado-Frias, "Hybrid Wave-Pipelined Network Router," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, December 2002, pp. 1764 - 1772.

[12] J. G. Delgado and J. Nyathi "A Wave-Pipelined Network Router," *IEEE Computer Society Workshop on VLSI 2001*, April 19-20, 2001, pp. 165-170.

[13] J. Nyathi, J. G. Delgado and J. Lowe, "A High Performance, Hybrid Wave-pipelined Linear Feedback Shift Register with Skew Tolerant Clocks" 46<sup>th</sup> IEEE International Midwest Symposium On Circuits and Systems, Cairo, Egypt, December 27-30, 2003.