

CLA and Ling Adders

1 Introduction

One of the most popular designs for fast integer adders are Carry-Look-Ahead adders. Rather than waiting for carry signals to ripple from the least significant bit to the most significant bit, CLA adders divided the inputs into groups of r bits and implement the logic equations to determine if each group will generate or propagate a carry. By combining the generate and propagate signals of r groups at with each successive stage of logic, a CLA adder can derive the carries into each bit in order $\log_r n$ gates instead of order n for a ripple carry adder. This paper discusses the design of a very simple 32 bit CLA adder, some improvements that can be made to that adder, and a variation of CLA adders known as Ling adders.

2 A Simple CLA Adder

An overview of the adder's 4 stages is shown in figure 1 with stage 1 and the top and stage 4 at the bottom. In stage 1 the local generate and propagate signals for each bit are created. In stage 2 these signals are combined to create generate and propagate signals out of each group of 3 bits. In stage 3 the group signals are combined into 9 bit block signals. In stage 4 the carry into each block is calculated and these signals begin traveling back up the adder tree. In stage 3 the carry into each group is created, and in stage 2 the carry into each bit is created. Finally, stage 1 uses the local carry signals to calculate the final sum bits.

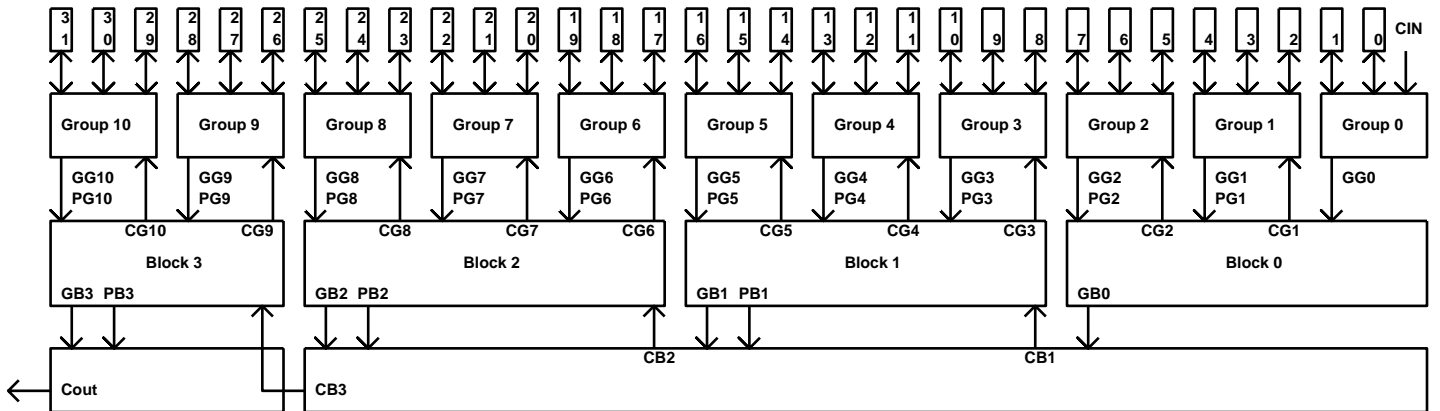


Figure 1: CLA Adder

2.1 Generate and Propagate Signals

In the first stage of logic the adder must calculate the local generate and propagate signals (g_i and p_i) which tell if each bit will generate a carry into the next bit or propagate a carry from the previous bit.

$$g_i = a_i b_i \quad (1)$$

$$p_i = a_i + b_i \quad (2)$$

In stage 2 these signals are then combined into group generates and propagates (GG_i and PG_i) for each of the ten groups as follows:

$$GG_0 = g_1 + p_1(g_0 + p_0 c_{IN}) \quad (3)$$

$$GG_1 = g_4 + p_1(g_3 + p_3 g_2) \quad (4)$$

$$\vdots$$

$$GG_{10} = g_{31} + p_{31}(g_{30} + p_{30} g_{29}) \quad (5)$$

$$PG_1 = p_4 p_3 p_2 \quad (6)$$

$$PG_2 = p_7 p_6 p_5 \quad (7)$$

$$\vdots$$

$$PG_{10} = p_{31} p_{30} p_{29} \quad (8)$$

where c_{IN} is the carry in signal to the least significant bit. Since c_{IN} is included in GG_0 , no group propagate signal from group 0 is needed. The group propagate signals are formed with a simple 3 input AND gate. The group generate signals are formed with the fanin-3 generate gate shown in figure 2. In stage 3 these signals are used to create the block generate and propagate signals (GB_i and PB_i).

$$GB_0 = GG_2 + PG_2(GG_1 + PG_1 GG_0) \quad (9)$$

$$GB_1 = GG_5 + PG_5(GG_4 + PG_4 GG_3) \quad (10)$$

$$GB_2 = GG_8 + PG_8(GG_7 + PG_7 GG_6) \quad (11)$$

$$GB_3 = GG_{10} + PG_{10} GG_9 \quad (12)$$

$$PB_1 = PG_5 PG_4 PG_3 \quad (13)$$

$$PB_2 = PG_8 PG_7 PG_6 \quad (14)$$

$$PB_3 = PG_{10} PG_9 \quad (15)$$

All the blocks can use the same fanin-3 generate gate and 3 input AND gate used in the previous stage except for block 3 which contains only two groups. Its propagate signal requires only a 2 input OR, and its generate is create using a fanin-2 generate gate shown in figure 3. Having created the block generate and propagate signals, the adder begins to finally create the true carry signals.

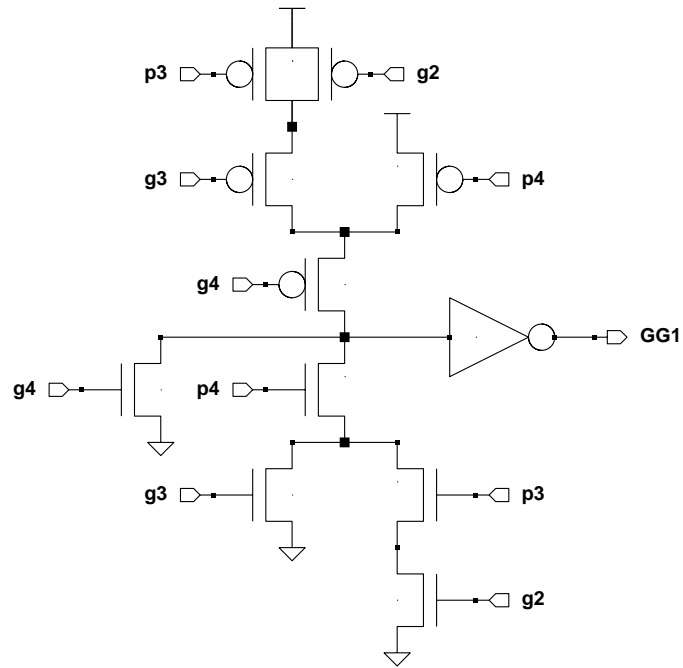


Figure 2: FanIn-3 Generate Gate

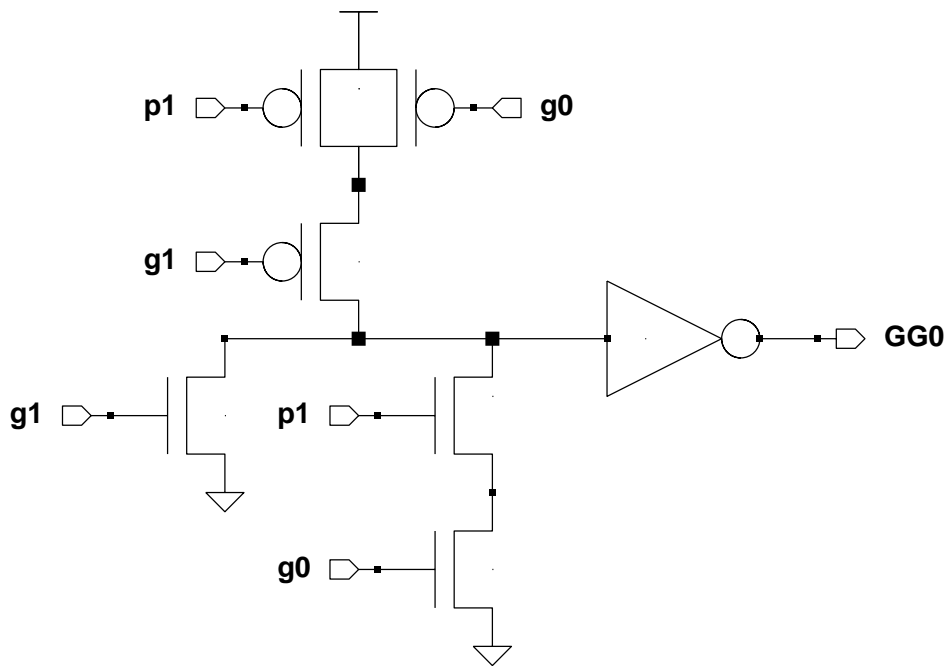


Figure 3: FanIn-2 Generate Gate

2.2 Carry Signals

In stage 4 the block generate and propagate signals are used to create the carry signals into each block (CB_i).

$$CB_1 = GB_0 \quad (16)$$

$$CB_2 = GB_1 + PB_1GB_0 \quad (17)$$

$$CB_3 = GB_2 + PB_2(GB_1 + PB_1GB_0) \quad (18)$$

$$C_{OUT} = GB_3 + PB_3CB_3 \quad (19)$$

where C_{OUT} is the overflow carry out of the entire adder. These signals then begin to travel back up the adder stages, first forming the carry into each group (CG_i) in stage 3. For block 2 these equations are as follows:

$$CG_6 = CB_2 \quad (20)$$

$$CG_7 = GG_6 + PG_6CB_2 \quad (21)$$

$$CG_8 = GG_7 + PG_7(GG_6 + PG_6CB_2) \quad (22)$$

In stage 2 these group carry signals are used to form the local carry into each bit (c_i). For group 8 these equations are as follows:

$$c_{23} = CG_8 \quad (23)$$

$$c_{24} = g_{23} + p_{23}CG_8 \quad (24)$$

$$c_{25} = g_{24} + p_{24}(g_{23} + p_{23}CG_8) \quad (25)$$

All of these signals can be created using the fanin-3 and fanin-2 generate gates shown in figures 2 and 3. This means each center group and block will use one fanin-3 gate and one OR gate to create generate and propagate signals for the stage below, and one fanin-3 gate and one fanin-2 gate to create carry signals for the stage above. The wiring of these groups and blocks is shown in figure 4 for group 1. When the local carry signals reach stage 1, they are used to create the final sum bits (s_i) according to the equations:

$$t_i = a_i \oplus b_i \quad (26)$$

$$s_i = t_i \oplus c_i \quad (27)$$

2.3 Critical Path

The worst case inputs for this adder are when $a_i \oplus b_i = 1$ for all the input bits and then c_{IN} is toggled. The local generate signals require 3 series transistors to form. For an N bit CLA adder combining r groups at each level, the generate signals must travel up $\lceil \log_r N \rceil - 1$ levels of $r + 1$ series transistors each. Then the signal travels down $\lceil \log_r N \rceil - 2$ levels of no more than $r + 1$

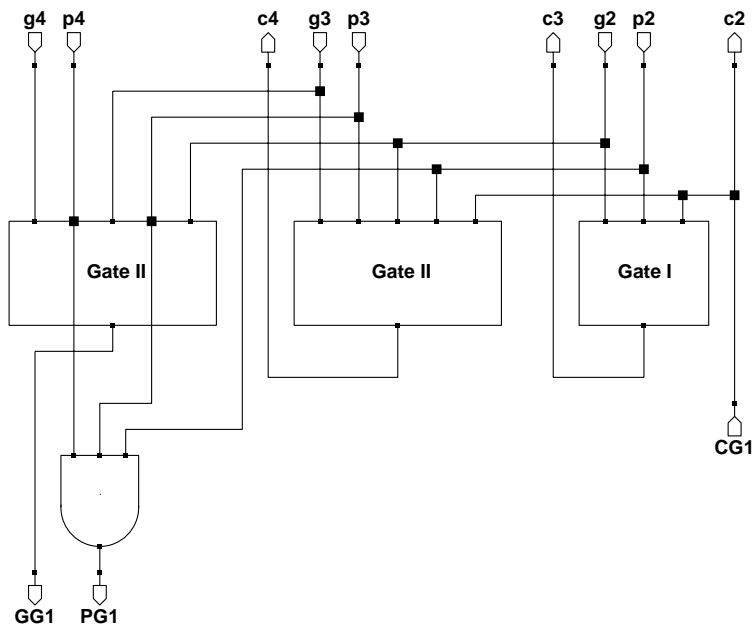


Figure 4: Group 1

series transistors. Final, the XOR to form the local sums takes 2 series transistors. Therefore, the maximum number of series transistors in the critical path can be written as:

$$T_d = 3 + (\lceil \log_r N \rceil - 1)(r + 1) + (\lceil \log_r N \rceil - 2)(r + 1) + 2 \quad (28)$$

$$T_d = (2 \lceil \log_r N \rceil - 3)(r + 1) + 5 \quad (29)$$

For a 32 bit adder with $r = 3$ as described in this paper this equation gives a maximum of 25 transistors. The true critical path is 24 transistors since block 3 contains only 2 groups instead of 3. The critical path is shown in table 1. Although faster designs are possible, this adder has the

Operation	Signal	Delay	Total
Local Generate	g_i	3	3
Group Generate	GG_i	4	7
Block Generate	GB_i	4	11
Block Carry	CB_3	4	15
Group Carry	GG_{10}	3	18
Local Carry	c_{31}	4	22
Local Sum	s_{31}	2	24

Table 1: Simple CLA Critical Path

advantage of a relatively simple layout and wiring. The next section discusses changes which can

be made in this design to improve performance.

3 An Improved CLA Adder

The critical path delay of the simple CLA adder design presented in the previous section can be reduced significantly at the price of making the layout and wiring more complex.

3.1 Single Stage Group Generate

The first improvement to be made is using a single complex gate to create the group generate and propagate signals in a single stage directly from the adder inputs. In the simple design the expression used for the group 1 generate signal was as follows:

$$GG_1 = g_4 + p_4(g_3 + p_3g_2) \quad (30)$$

Expanding this in terms of the adder inputs gives:

$$GG_1 = a_4b_4 + (a_4 + b_4)[a_3b_3 + (a_3 + b_3)a_2b_2] \quad (31)$$

This equation can be implemented by an NMOS network containing 4 series transistors followed by an inverter. The PMOS network must implement the complement of this function, which normally would also require 4 series transistors. However, the relation $\overline{g_i p_i} = \overline{p_i}$ can be used to simplify the expression for $\overline{GG_1}$ as follows:

$$\overline{GG_1} = \overline{g_4}[\overline{p_4} + \overline{g_3}(\overline{p_3} + \overline{g_2})] \quad (32)$$

$$\overline{GG_1} = \overline{p_4} + \overline{g_4}(\overline{p_3} + \overline{g_3} \overline{g_2}) \quad (33)$$

$$\overline{GG_1} = \overline{a_4} \overline{b_4} + (\overline{a_4} + \overline{b_4})[\overline{a_3} \overline{b_3} + (\overline{a_3} + \overline{b_3})(\overline{a_2} + \overline{b_2})] \quad (34)$$

This simplified expression can be implemented by a PMOS network with 3 series transistors followed by an inverter. The gate implementing the group generate for group 1 is shown in figure 5. The gate implementing the group propagate is shown in figure 6. This change reduces the total number of series transistors used in forming the group generate signals from 7 to 5.

3.2 Carry Select Mux

The second improvement eliminates the need to travel back up the adder tree after the block carries have been formed. This is done by generating two sets of sum bits. One set assumes the carry into each block will be 0, and the other set assumes it will be 1. This can occur in parallel with the generation of the block carries which are then used to control a mux which selects the proper set of sum bits. This is the same method used in carry select adders.

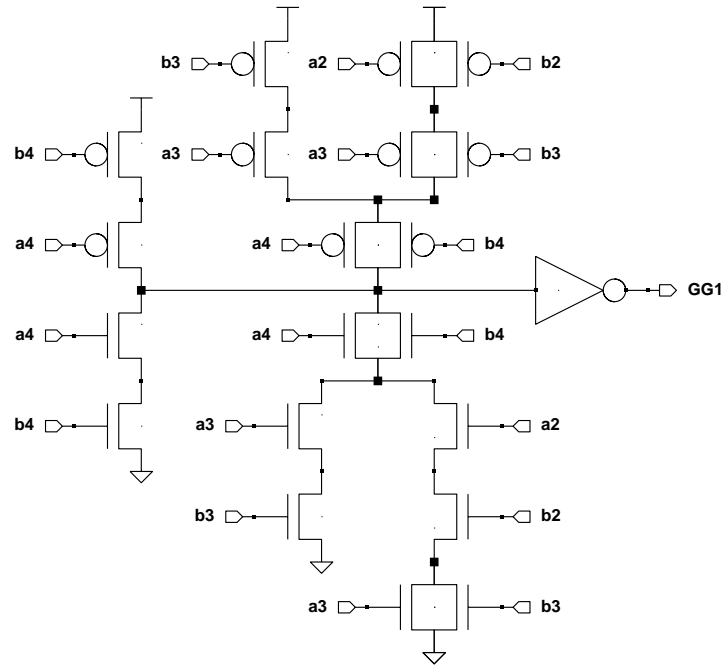


Figure 5: CLA Group Generate

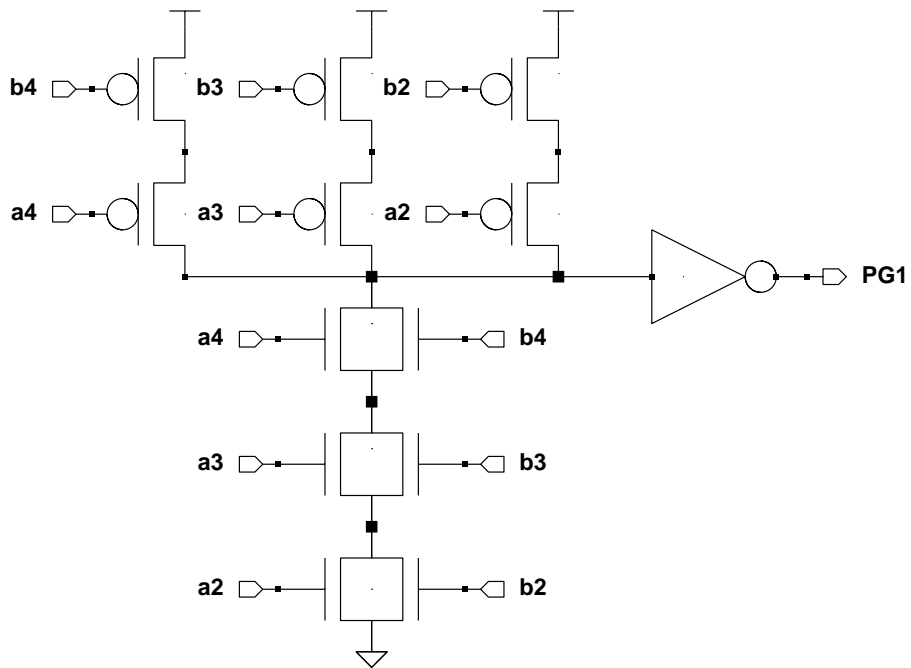


Figure 6: CLA Group Propagate

In the simple CLA adder the equations implemented by group carry, local carry, and final sum stages for bit 23 are as follows:

$$s_{23} = t_{23} \oplus c_{23} \quad (35)$$

$$s_{23} = t_{23} \oplus CG_8 \quad (36)$$

$$s_{23} = t_{23} \oplus [GG_7 + PG_7(GG_6 + PG_6CB_2)] \quad (37)$$

This expression is converted to a mux controlled by CB_2 by defining the signals CGF_8 and CGT_8 :

$$CGF_8 = GG_7 + PG_7GG_6 \quad (38)$$

$$CGT_8 = GG_7 + PG_7(GG_6 + PG_6) \quad (39)$$

The signal CGF_8 is the carry into group 8 assuming the block carry is zero, and CGT_8 assumes the block carry is one. The final sum bit is then written as:

$$s_{23} = \overline{CB_2}[CGF_8 \oplus t_{23}] + CB_2[CGT_8 \oplus t_{23}] \quad (40)$$

Using these signals, the other sum bits of the group are written in similar fashion.

$$s_{24} = \overline{CB_2}[(g_{23} + p_{23}CGF_8) \oplus t_{24}] + CB_2[(g_{23} + p_{23}CGT_8) \oplus t_{24}] \quad (41)$$

$$s_{25} = \overline{CB_2}[(g_{24} + p_{24}(g_{23} + p_{23}CGF_8)) \oplus t_{24}] + CB_2[(g_{24} + p_{24}(g_{23} + p_{23}CGT_8)) \oplus t_{24}] \quad (42)$$

Because the signals CGF_8 and CGT_8 will appear after the local generate and propagate signals, the critical path delay can be further reduced by applying the same principal to make the inputs to the mux controlled by the block carry muxes controlled by CGF_8 and CGT_8 . This also allows the simplification of $g_i + p_i = p_i$ to be applied.

$$s_{23} = \overline{CB_2}[\overline{CGF_8}t_{23} + CGF_8\overline{t_{23}}] + CB_2[\overline{CGT_8}t_{23} + CGT_8\overline{t_{23}}] \quad (43)$$

$$s_{24} = \overline{CB_2}[\overline{CGF_8}(g_{23} \oplus t_{24}) + CGF_8(p_{23} \oplus t_{24})] + CB_2[\overline{CGT_8}(g_{23} \oplus t_{24}) + CGT_8(p_{23} \oplus t_{24})] \quad (44)$$

$$s_{25} = \overline{CB_2}\{\overline{CGF_8}[(g_{24} + p_{24}g_{23}) \oplus t_{25}] + CGF_8[(g_{24} + p_{24}p_{23}) \oplus t_{25}]\} + CB_2\{\overline{CGT_8}[(g_{24} + p_{24}g_{23}) \oplus t_{25}] + CGT_8[(g_{24} + p_{24}p_{23}) \oplus t_{25}]\} \quad (45)$$

The 3 bit slice which implements these functions is shown for group 8 in figure 7. Using the bit slice eliminates the need to go back up the adder tree after forming the block carries, and reduces the critical path after the block carries to a single mux delay.

Because of the reduced delay from the formation of the block carries to the final sum output, C_{OUT} can no longer be implemented as a function of CB_2 as shown in equation 19 without becoming the critical path. To avoid this a fanin-4 generate gate is used to form C_{OUT} directly from the block generates and propagates.

$$C_{OUT} = GB_3 + PB_3[GB_2 + PB_2(GB_1 + PB_1GB_0)] \quad (46)$$

This gate is shown in figure 8 and removes C_{OUT} from the critical path.

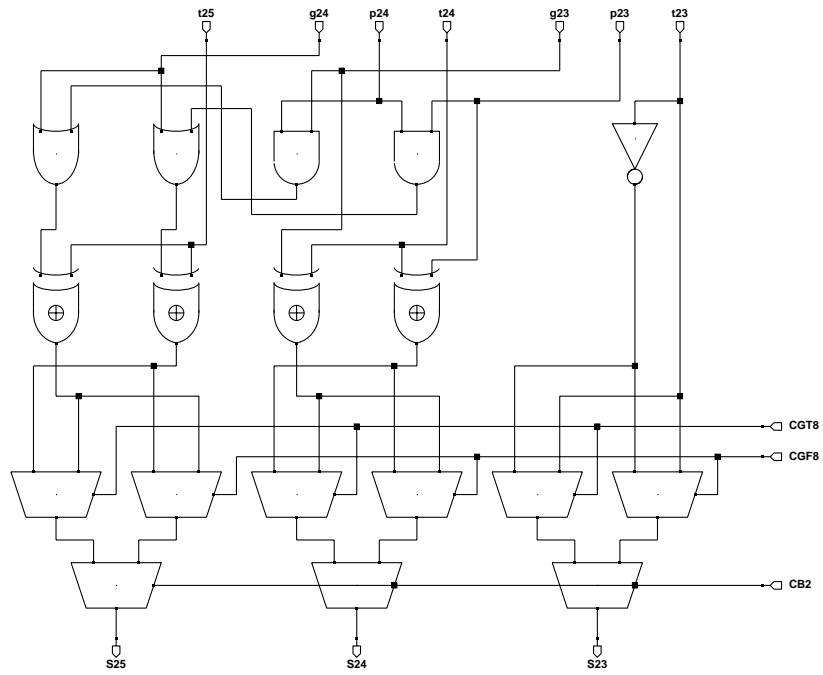


Figure 7: Sum Selection Slice

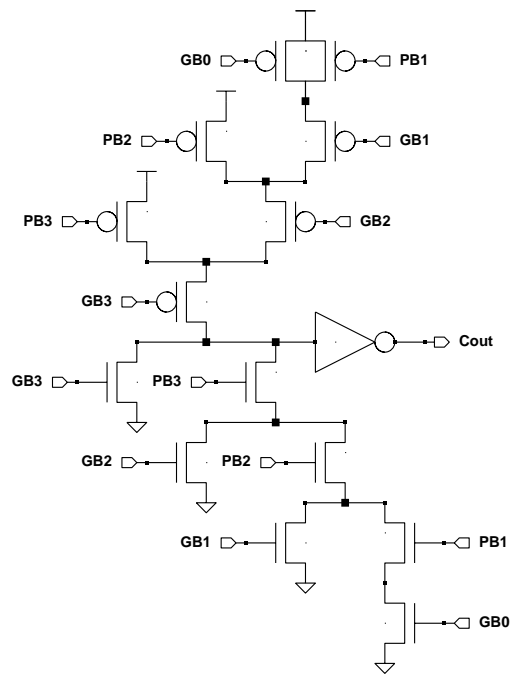


Figure 8: FanIn-4 Generate Gate

3.3 Critical Path

With a single stage group generate the critical path must still pass up $\lceil \log_r N \rceil - 1$ levels. Of these the first level will contain $r + 2$ series transistors and the others $r + 1$. The carry select mux eliminates the need to travel back up the levels of the adder to form the local carries. The mux delay from the arrival of the control signal is counted as one series transistor to form the complement of the control signal and one transistor to pass the input to the output. The number of series transistors in the critical path is therefore:

$$T_d = (\lceil \log_r N \rceil - 1)(r + 1) + 3 \quad (47)$$

For the 32 bit adder shown here with $r = 3$ this gives 15 series transistors. Using the single stage group generate eliminates 2 series transistors, and the carry select mux reduces the delay from the formation of the block carries from 9 series transistors to 2. The total critical path is reduced by 9 series transistors from a total of 24 to 15. The new critical path is shown in table 2.

Operation	Signal	Delay	Total
Group Generate	GG_i	5	5
Block Generate	GB_2	4	9
Block Carry	CB_3	4	13
Result Mux	s_{31}	2	15

Table 2: Improved CLA Critical Path

4 A Ling Adder

One final improvement that can be made to CLA design is the use of a pseudo-carry as proposed by Ling[1, 2]. This method allows a single local propagate signal to be removed from the critical path. To show how this is done the group generate signal for group 1 is shown below:

$$GG_1 = g_4 + p_4g_3 + p_4p_3g_2 \quad (48)$$

Ling observed that each term in GG_1 contains p_4 except for the very first term which is simply g_4 . However, p_4 can still be factored out of this expression by noting that $g_i = p_i g_i$.

$$GG_1 = p_4 GG_1^* \quad (49)$$

$$GG_1^* = g_4 + g_3 + p_3p_2 \quad (50)$$

The Ling group generate signal (GG_1^*) is simpler and can be calculated more quickly than the CLA group generate signal. When expanded out the CLA and Ling group generates are as follows:

$$GG_1 = a_4b_4 + (a_4 + b_4)[a_3b_3 + (a_3 + b_3)a_2b_2] \quad (51)$$

$$GG_1^* = a_4b_4 + a_3b_3 + (a_3 + b_3)a_2b_2 \quad (52)$$

The gate used to implement the group generate signal is shown in figure 9 and has one less series transistor than the equivalent CLA gate shown in figure 5. The Ling group propagate signals (PG_i^*)

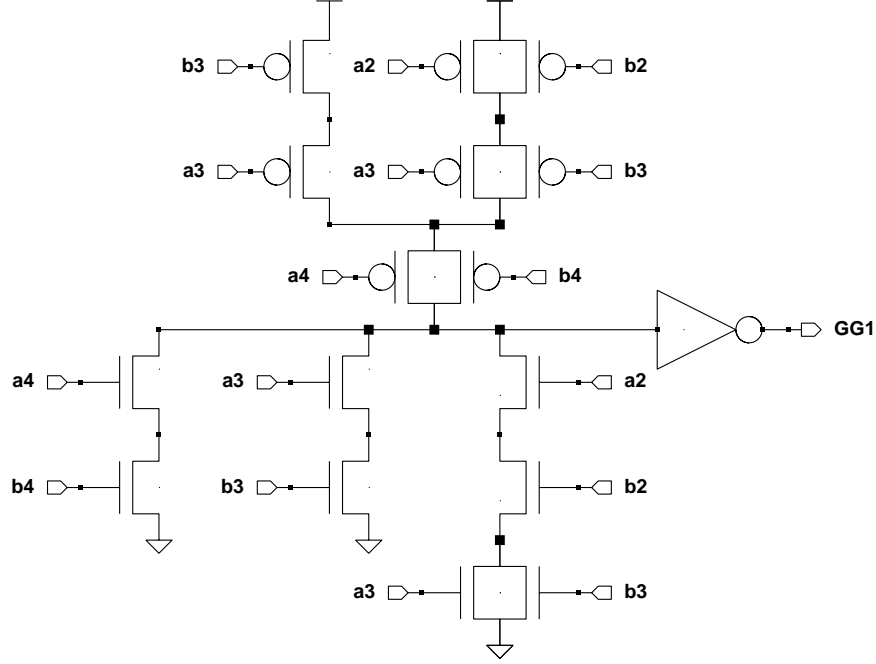


Figure 9: Ling Group Generate

are formed using the same gates as in the CLA design, but they are shifted one bit to the right. The CLA and Ling group propagate signals for group one are shown below.

$$PG_1 = p_4 p_3 p_2 \quad (53)$$

$$PG_1^* = p_3 p_2 p_1 \quad (54)$$

These Ling group generate and propagate signals are then combined in the same manner as before to create block carry signals.

$$CB_1^* = GB_0^* \quad (55)$$

$$CB_2^* = GB_1^* + PB_1^* GB_0^* \quad (56)$$

$$CB_3^* = GB_2^* + PB_2^* (GB_1^* + PB_1^* GB_0^*) \quad (57)$$

$$C_{OUT}^* = GB_3^* + PB_3^* [GB_2^* + PB_2^* (GB_1^* + PB_1^* GB_0^*)] \quad (58)$$

The true C_{OUT} is simply $p_{31} C_{OUT}^*$ which could be formed with a simple AND gate, but this would make it the critical path. Instead, the final group generate signal (GG_{10}) is formed using the CLA expression rather than the Ling group generate. Also the final group propagate (PG_{10}^*) is formed with a 4 input AND instead of a 3 input AND to include p_{31} . These changes allow the true C_{OUT} to be formed from the block generate and propagate signals as shown above without making it the critical path.

The final change that must be implemented to complete the Ling adder is to insert into the sum logic the local propagate signal which was factored out of each group generate. This is done simply by ANDing the CGF_i^* and CGT_i^* signals formed from the Ling group generate and propagates with the local propagate signal of the most significant bit of the previous group. This change is shown in figure 10 which depicts the sum selection logic for group 8 of the Ling adder.

4.1 Critical Path

The only difference in the critical path of the improved CLA and the Ling adder is the use of the Ling group generate is the first stage as shown in table 3. This allows the group generate signals to be formed in $r + 1$ series transistors instead of $r + 2$. The changes in the sum selection logic are off the critical path and have no effect on the total delay. Therefore, the series transistors in the critical path can be written as:

$$T_d = (\lceil \log_r N \rceil - 1)(r + 1) + 2 \quad (59)$$

For a 32 bit adder with $r = 3$ the net improvement of a Ling adder over the improved CLA adder is a total delay of 14 series transistors instead of 15.

Operation	Signal	Delay	Total
Group Generate	GG_i	4	4
Block Generate	GB_2	4	8
Block Carry	CB_3	4	12
Result Mux	s_{31}	2	14

Table 3: Ling Critical Path

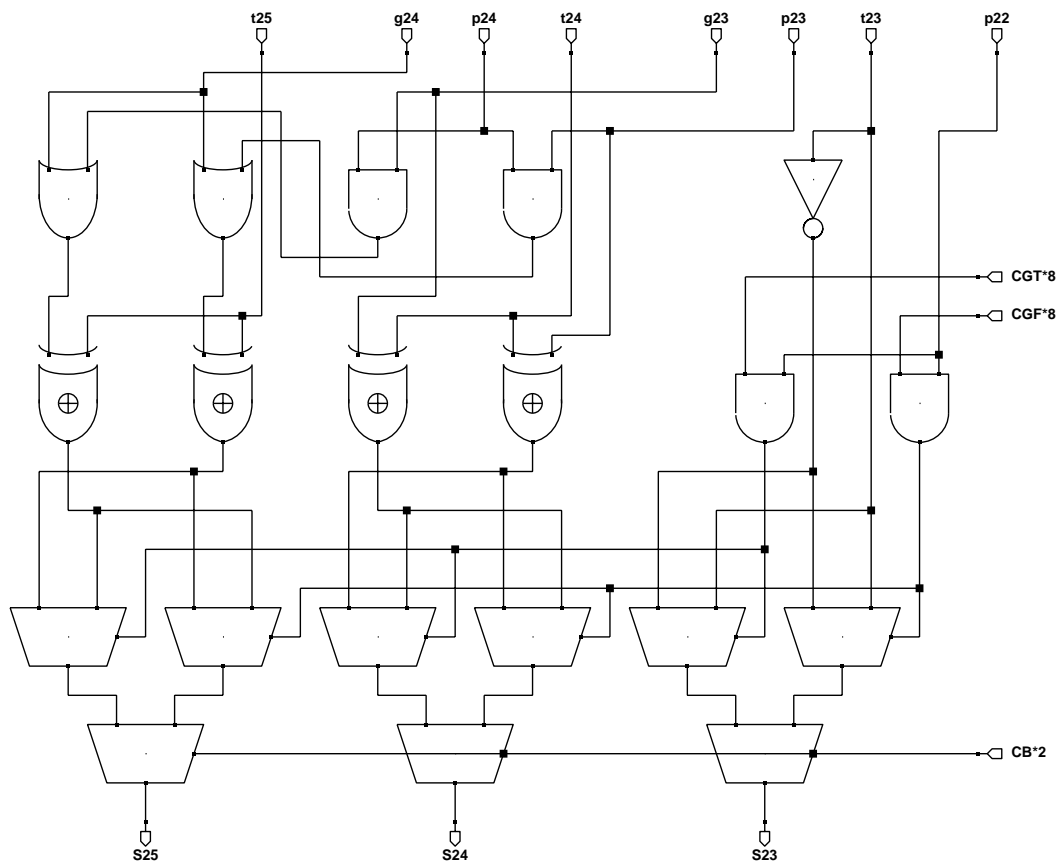


Figure 10: Ling Sum Selection Slice

References

- [1] H. Ling. High speed binary parallel adder. *IEEE Transactions on Computers*, EC-15(5):799–802, October 1966.
- [2] H. Ling. High speed binary adder. *IBM Journal of Research and Development*, 25(3):156–166, May 1981.
- [3] R. Brent and H. Kung. A regular layout for parallel adders. *IEEE Transactions on Computers*, C-31(3):260–264, March 1982.
- [4] G. Bewick, P. Song, G. DeMicheli, and M. Flynn. Approaching a nanosecond: A 32-bit adder. In *Proceedings of the International Conference on Computer Design*, pages 221–224, 1988.
- [5] I. Hwang and A. Fisher. A 3.1ns 32b CMOS adder in multiple output domino logic. In *International Solid State Circuits Conference*, pages 140–141, 1988.
- [6] A. Omondi. *Computer Arithmetic Systems: Algorithms, Architecture and Implementations*. Prentice Hall, 1994.
- [7] N. Quach and M. Flynn. High-speed addition in CMOS. Technical Report CSL-TR-90-415, Stanford University, February 1990.
- [8] S. Waser and M. Flynn. *Introduction to Arithmetic for Digital Systems Designers*. Holts, Rinehart and Winston, 1982.