Logarithmic 16 bit adder using Brent Kung architecture

Self Project

BY

P.Nagasai Goud 203070094

DEPARTMENT OF ELECTRICAL ENGINEERING

IIT BOMBAY
Main Gate Rd, IIT Area, Powai, Mumbai, Maharashtra 400076

Tree Adders

- Tree adders use the idea of carry look ahead addition.
- However, these do not try to implement the complex logic expressions which result from looking ahead all the way.
- Instead, these build up the logic in a tree like structure, where each node performs simple logic operations on the results of the previous node.
- Because of the tree structure used in this, the delay is of the O(logn) for an n bit adder.

For Carry Look Ahead, we had defined

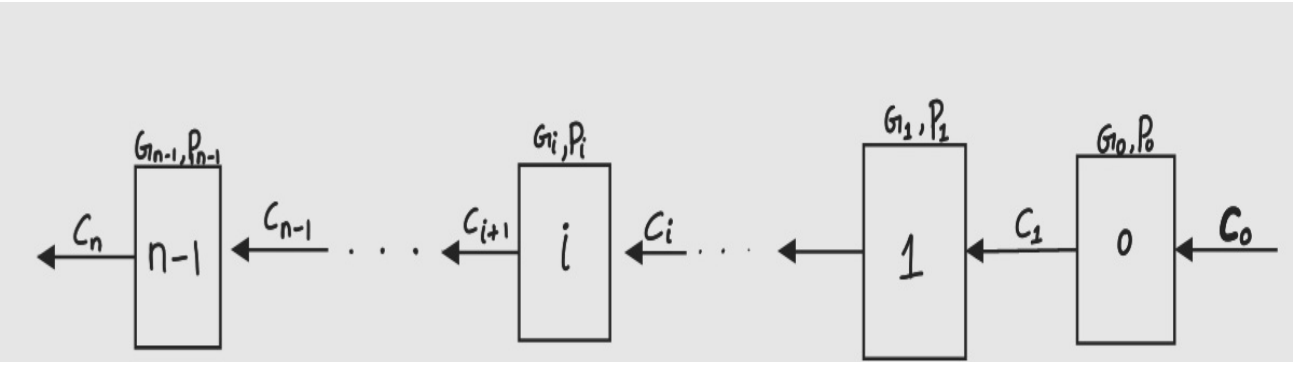
$K = \overline{A}.\overline{B},\ \ G = A.B\ \ and\ \ P = A \oplus B$

P, G and K can be computed without waiting for Cin.

when K = 1 C_{out} = 0 irrespective of C_{in}.

when G = 1 C_{out} = 1 irrespective of C_{in}.

when P = 1 $C_{out} = C_{in} \implies$ This is the only case when we must wait for C_{in} in order to compute C_{out}



The least significant bit is indexed as 0 and the most significant bit as $n-1$

i'th bit accepts C_i as input carry and produces C_{i+1} as output carry. Output of the i'th stage is:

$$C_{i+1} \;=\; G_i \;+\; P_i.C_i \tag{1}$$

C_i itself is the output of cell no. (i-1). Hence,

$$C_{i+1} = G_i + P_i.(G_{i-1} + P_{i-1}.C_{i-1}) \implies (G_i + P_i.G_{i-1}) + (P_i.P_{i-1}).C_{i-1}$$

$$C_{i+1} = G^2_{i,i-1} + P^2_{i,i-1}.C_{i-1} \tag{2}$$

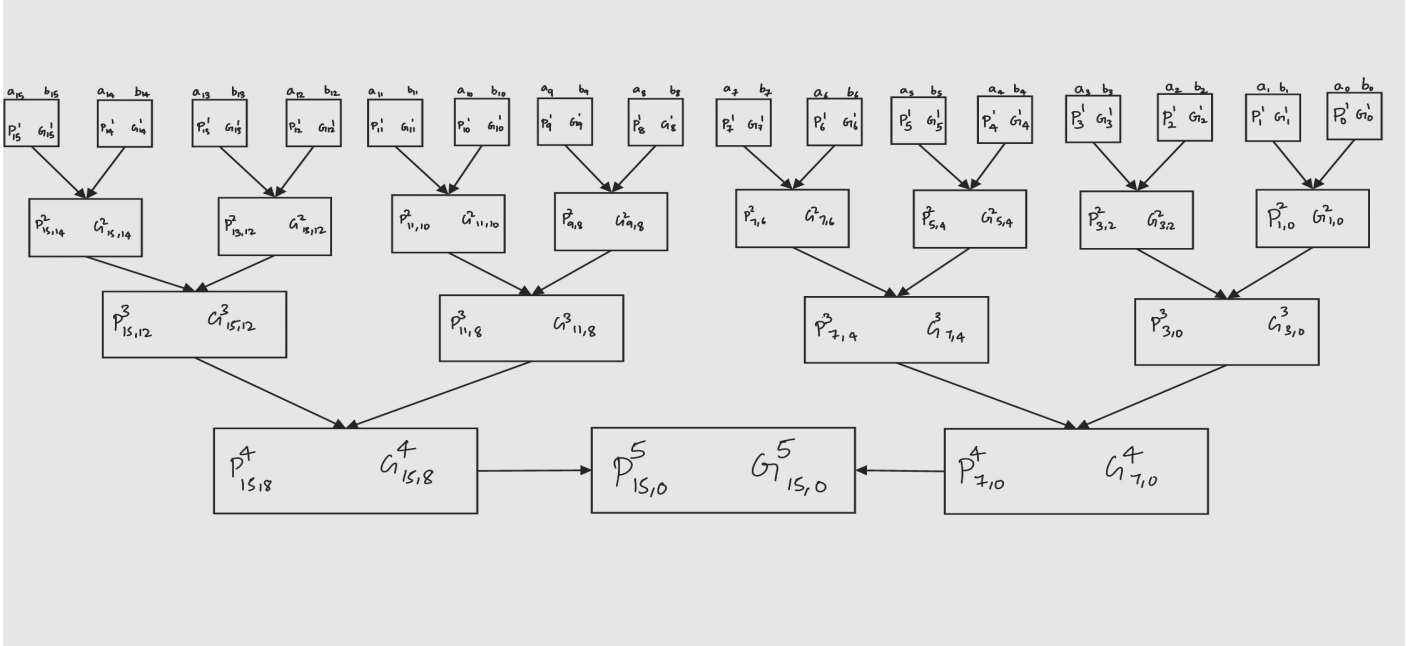
which evaluates C_{i+1} directly from C_{i-1}.

- Once the highest order P and G values have been generated, the final carry can be computed in one step from the input carry.
- In principle, we do not need the internal carries at each bit for the final result.
- However, addition is not complete unless all the sum bits have also been generated.

  The sum bits are given by: $S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$

Brent Kung Adder

- The Brent Kung tree adder is a logarithmic adder of low complexity.
- Values of P and G are computed in a tree fashion
- The figure below shows the generation of P and G values for an 16 bit adder.



we first calculate P_i^1, G_i^1, with i = 0 15

$$G_i^1 = A_i.B_i, \;\; P_i^1 = A_i \oplus B_i$$

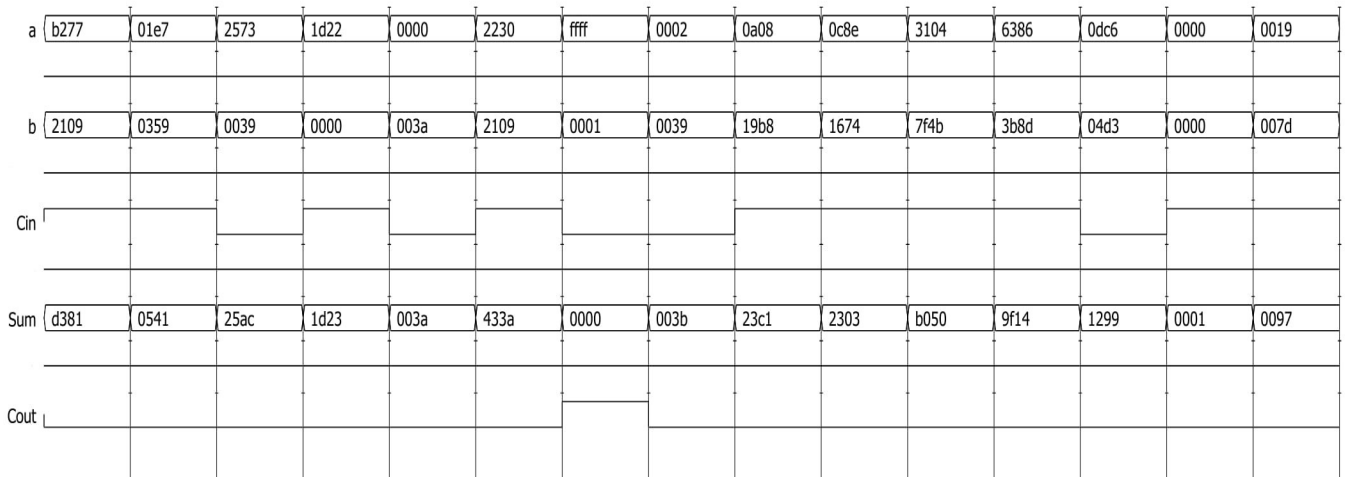
Next, using these values, we can generate $P^2_{2i+1,2i}, G^2_{2i+1,2i}$, with i = 0. . .7

$$G^2_{2i+1,2i} = G^1_{2i+1} + P^1_{2i+1}.G^1_{2i} \;\; , \;\; P^2_{2i+1,2i} = P^1_{2i+1}.P^1_{2i}$$

Similarly we can find all other higher order G,P values.

After finding all orders of G,P then we can find the carry from equation (1),(2)

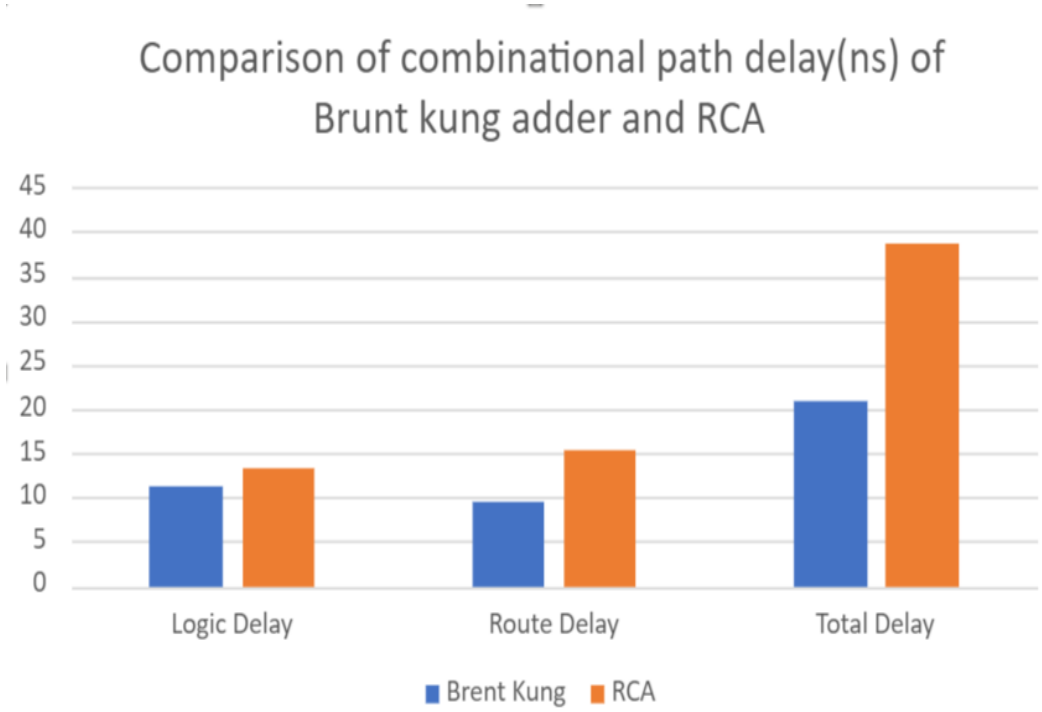
With all carry values generated, the corresponding sum values can be calculated using the relation $Sum_i = P_i^1 \oplus C_i$




```
#                  0 << Starting the Simulation >>
# time=                0 A = 45687, B =  8457, Cin= 1 : Sum = 54145, Cout = 0
#
# time=               20 A =   487, B =   857, Cin= 1 : Sum =  1345, Cout = 0
#
# time=               40 A =  9587, B =    57, Cin= 0 : Sum =  9644, Cout = 0
#
# time=               60 A =  7458, B =     0, Cin= 1 : Sum =  7459, Cout = 0
#
# time=               80 A =     0, B =    58, Cin= 0 : Sum =    58, Cout = 0
#
# time=              100 A =  8752, B =  8457, Cin= 1 : Sum = 17210, Cout = 0
#
# time=              120 A = 65535, B =     1, Cin= 0 : Sum =     0, Cout = 1
#
# time=              140 A =     2, B =    57, Cin= 0 : Sum =    59, Cout = 0
#
# time=              160 A =  2568, B =  6584, Cin= 1 : Sum =  9153, Cout = 0
#
# time=              180 A =  3214, B =  5748, Cin= 1 : Sum =  8963, Cout = 0
#
# time=              200 A = 12548, B = 32587, Cin= 1 : Sum = 45136, Cout = 0
#
# time=              220 A = 25478, B = 15245, Cin= 1 : Sum = 40724, Cout = 0
#
# time=              240 A =  3526, B =  1235, Cin= 0 : Sum =  4761, Cout = 0
#
# time=              260 A =     0, B =     0, Cin= 1 : Sum =     1, Cout = 0
#
# time=              280 A =    25, B =   125, Cin= 1 : Sum =   151, Cout = 0
#
# time=              300 A =   852, B =     8, Cin= 1 : Sum =   861, Cout = 0
```

Figure 1: Adder output for Appropriate test vectors

	Logic Delay	Route Delay	Total Delay
Brent Kung	11.268ns	9.505ns	20.773ns
RCA	13.288ns	15.453ns	28.741ns



These delays are calculated in Xilinx Software.

Family :- Spartan-3 Device :- XC3s200

Package :- FT256 Speed : -5

Reference

- EE 671 Lectures IIT-Bombay by Prof.K.Dinesh Sharma