Indian Institute of Technology Kharagpur

Computer Organization Laboratory
Assignment-3
CS39001

Part 2:- Carry Look-ahead Adder

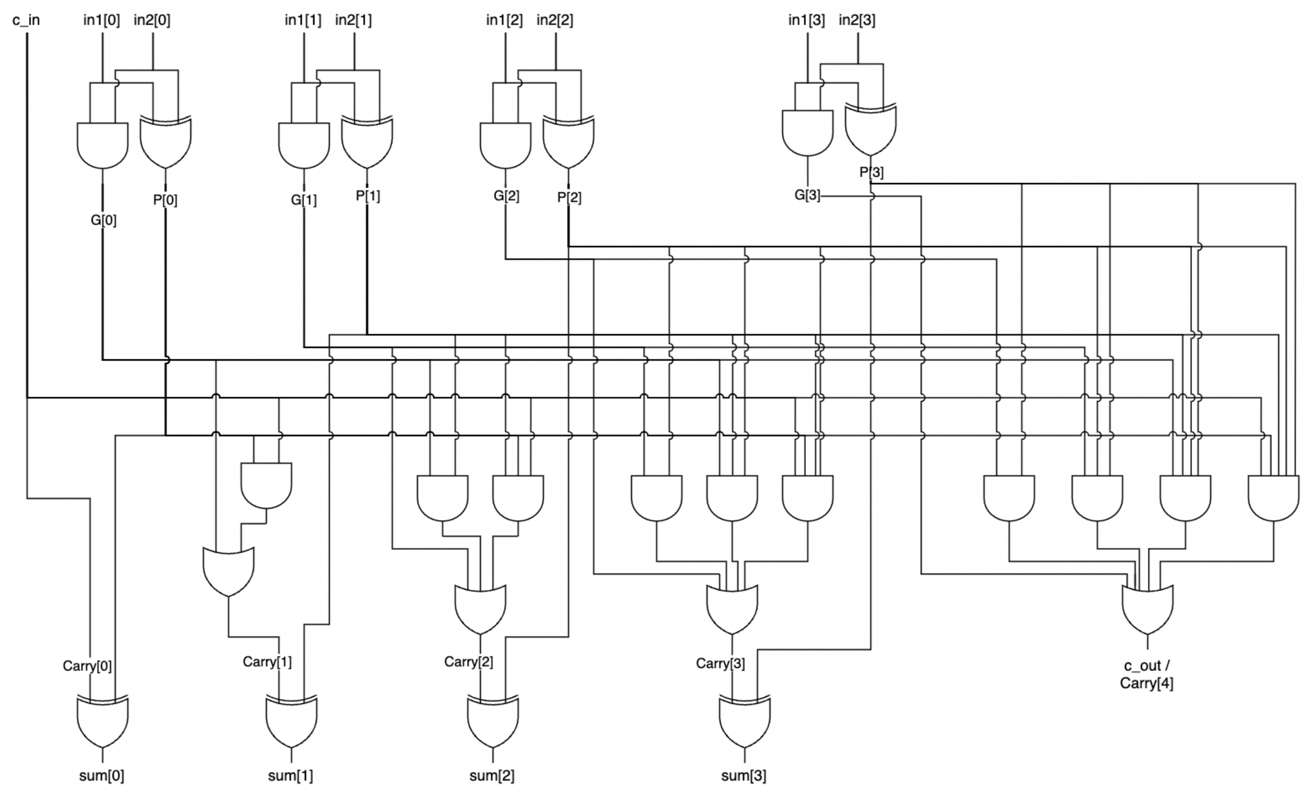
Hardik Pravin Soni 20CS30023
Abhay Kumar Keshari 20CS10001

Contents

1 4-bit Carry Look-ahead Adder(CLA)

The Carry Look-ahead adder decreases the latency in computing the sum of two integers by concurrently calculating the carries rather than waiting for the carry from the preceding block to ripple, as is the case with the ripple carry adder. The carry look-ahead adder's logic and circuit diagram are as follows:

1.1 Circuit Diagram



1.2 Logic Expression

Logic of the Circuit:-

$$P = a \oplus b$$

$$G = a \;\&\; b$$

Expanding for i,

$$P[i] = a[i] \oplus b[i] \textbf{ for all 0¡=i¡=3} \tag{1}$$

$$G[i] = a[i] \;\&\; b[i] \textbf{ for all 0¡=i¡=3} \tag{2}$$

Initialize C[0] to be c_{in} Then Recursively we can elaborate:-

$$s[i] = P[i] \oplus C[i] \;\; \text{0¡=i¡=3} \tag{3}$$

$$C[i] = G[i-1]|(P[i-1] \;\&\; C[i-1]) \text{ for all 1¡=i¡=4} \tag{4}$$

$$C0 = c_{in} \tag{5}$$

$$C1 = G[0]|(P[0] \;\&\; c_{in}) \tag{6}$$

$$C2 = G[1]|(P[1] \;\&\; C1) = G[1]|(P[1] \;\&\; (G[0]|(P[0] \;\&\; c_{in}))) = G[1]|(P1 \;\&\; G[0])|(P[1] \;\&\; P[0] \;\&\; c_{in}) \tag{7}$$

$$C3 = G[2]|(P[2] \;\&\; C2) = G[2]|(P[2] \;\&\; G[1])|(P[2] \;\&\; P[1] \;\&\; G[0])|(P[2] \;\&\; P[1] \;\&\; P[0] \;\&\; c_in) \tag{8}$$

$$C4 = G[3]|(P[3] \;\&\; C3) \tag{9}$$

Evaluating it further we get,

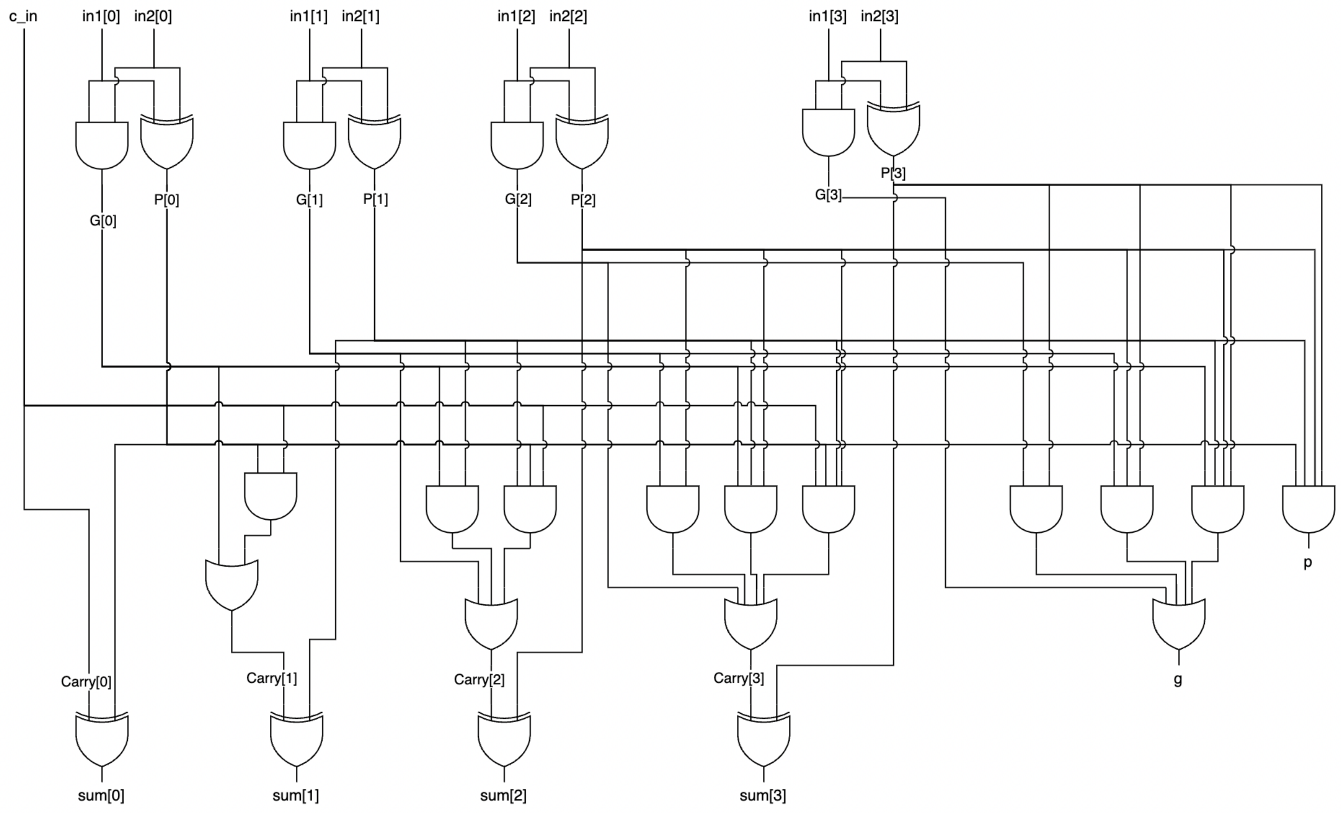
$$C4 = G[3]|(P[3] \;\&\; G[2])|(P[3] \;\&\; P[2] \;\&\; G[1])|(P[3] \;\&\; P[2] \;\&\; P[1] \;\&\; G[0])|(P[3] \;\&\; P[2] \;\&\; P[1] \;\&\; P[0] \;\&\; c_{in}) \tag{10}$$

$$c_{out} = C4 \tag{11}$$

2 4-bit Carry Look-ahead Adder(CLA) (augmented)

Instead of creating the *cout*, the 4-bit Carry Look-ahead Adder (augmented) provides the block propagate(P) and generate(G) as a signal, which is then used by the look carry ahead unit. We may create 16-bit, 32-bit, and 64-bit CLA Augmented Adders utilizing the block propagate(P) and generate(G) signals in this modular logic design.

2.1 Circuit Diagram



2.2 Logic Expression

The Block Propagate and the Generate Signals are calculated as follows:-

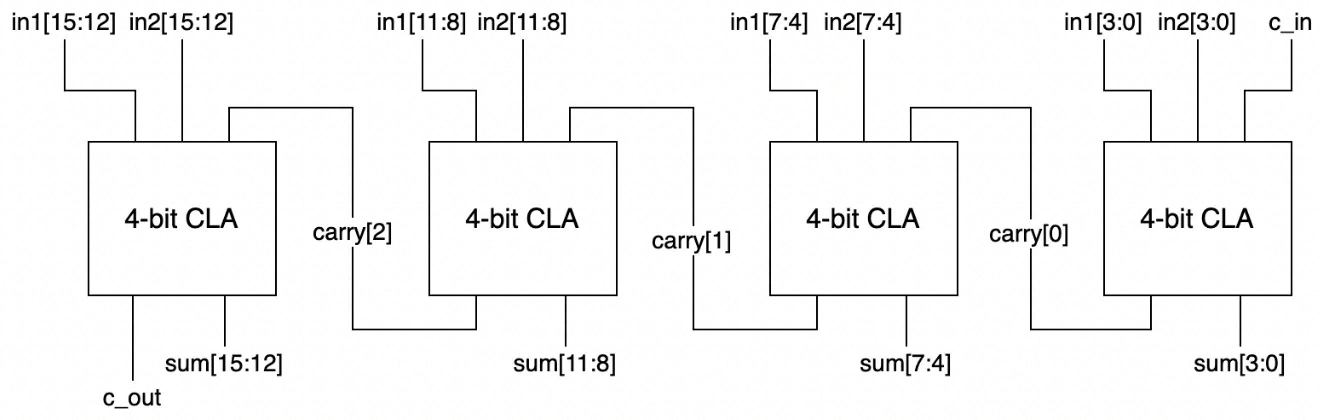
$$p = P[3] \ \& \ P[2] \ \& \ P[1] \ \& \ P[0] \tag{12}$$

$$g = G[3] | (P[3]\ \&\ G[2]) | (P[3]\ \&\ P[2]\ \&\ G[1]) | (P[3]\ \&\ P[2]\ \&\ P[1]\ \&\ G[0])) \tag{13}$$

3 16-bit Carry Look-ahead Adder(CLA) with Ripple Carry Out

The 16-bit Carry Look-ahead adder is built by cascading four 4-bit CLAs together. As illustrated in the graphic below, the carry out is rippling from one block to the next.

3.1 Circuit Diagram



Here carry[2:0] are internal queries carried out and being rippled as temporary, from one block to another.

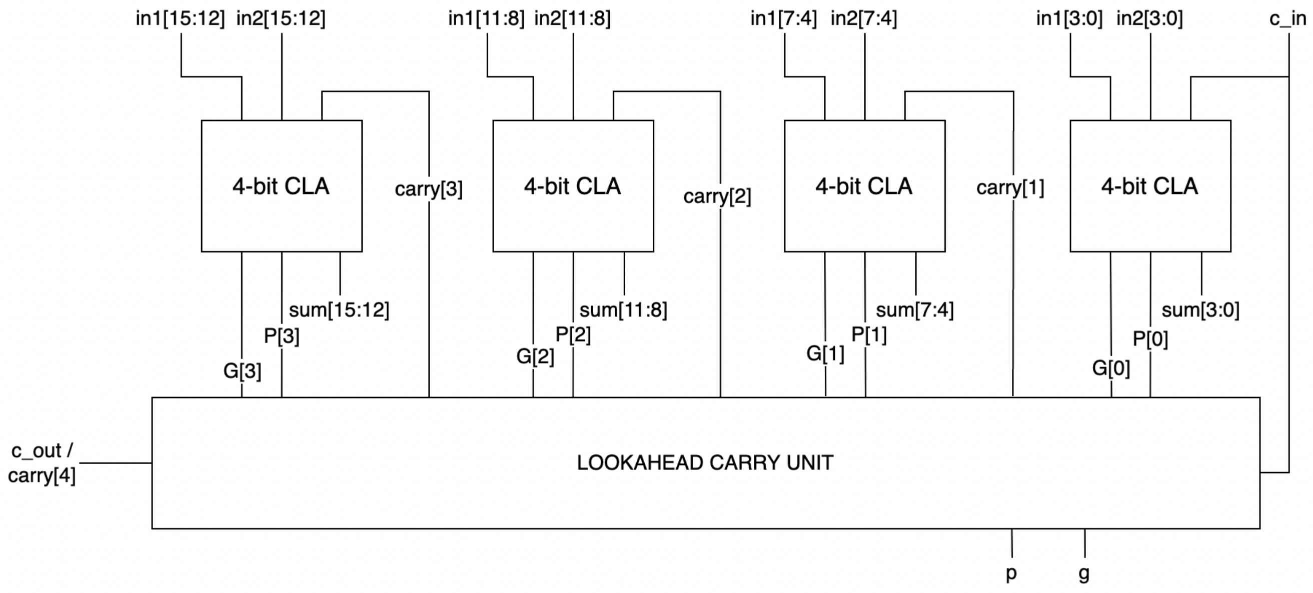
4 16-bit Carry Look-ahead Adder(CLA) with Look Ahead Carry Unit

$P[3:0]$: **–Block Propagate Signal**

$G[3:0]$: **–Generate Signal**

Instead of spreading out the carry out from one block to the next, we can now decrease the delay in the circuit by computing these requests concurrently. As a result, following blocks do not have to wait for the prior block's carry. As a result, a more modular architecture is produced that calculates the Block Propagate and Generate Signals, which are useful for creating CLA Adders of Higher Order.

4.1 Circuit Diagram



4.2 Logic Expression

The Logical Expressions for look-ahead carry unit are as follows:- for P, G and C0,C1,.. C4 we can look upon the logical expressions enlisted in the 4-bit look-ahead carry adder. for calculation of Block Propagate and Generate we have:-

$$p = P[3]\ \&\ P[2]\ \&\ P[1]\ \&\ P[0] \tag{14}$$

$$g = G[3]|(P[3]\ \&\ G[2])|(P[3]\ \&\ P[2]\ \&\ G[1])|(P[3]\ \&\ P[2]\ \&\ P[1]\ \&\ G[0])) \tag{15}$$

5 Synthesis Table

5.1 Comparison of 16-bit CLA Adder using Ripple Carry Out vs. Look-ahead Carry Unit

16-bit CLA with second level look-ahead carry unit performs better than rippling the carry at the cost of additional LUT's.

Table 1: 16-bit CLA Adder Ripple C_{out} vs. LCU

	Ripple carry out	**Look-ahead carry unit**
Delay(in ns)	2.784 ns	2.736 ns
Number of Slice LUT's	34	50
Number of Bonded IOB's	52	54
Levels of Logic	10	9

5.2 Comparison with 4-bit RCA

From the observations mentioned in the table below we can infer that 4-bit CLA is (almost 2.5x) faster than 4-bit RCA. Also the the Number of Slice LUT's is almost 1/3 rd.

Table 2: Comparison of 4-bit RCA vs. LCA

4-bit	**Delay(in ns)**	**Number of Slice LUT's**	**Logic Levels**	**Number of Bonded IOB's**
CLA	1.556 ns	6	3	16
RCA	4.005 ns	20	19	16

5.3 Comparison with 16-bit RCA

From the Observations mentioned in the table belo we can infer that 16-bit CLA is (almost 5.0x) faster than 16-bit RCA. Also the number of LUT's used are half as that of 16-bit RCA.

Table 3: Comparison of 16-bit RCA vs. LCA

16-bit	**Delay(in ns)**	**Number of Slice LUT's**	**Logic Levels**	**Number of Bonded IOB's**
CLA	2.736 ns	50	9	54
RCA	13.466 ns	80	70	50