# ASSIGNMENT 3 REPORT

*Carry Look-ahead Adder (4 and 16 bit)*

**Jatin Gupta (20CS10087)**

**Rushil Venkateswar (20CS30045)**

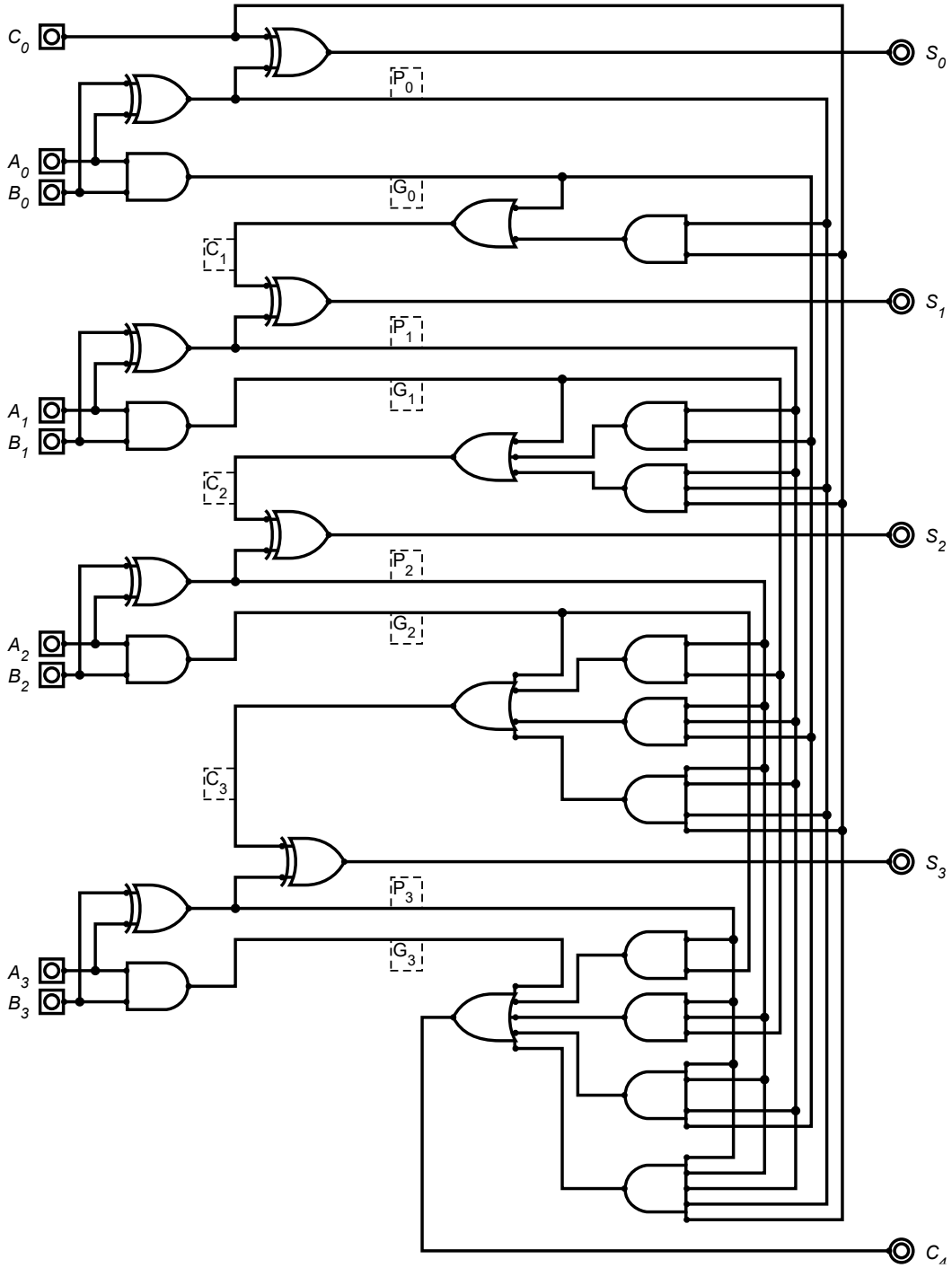
Group 69

Computer Organisation Laboratory

# CARRY LOOK-AHEAD ADDER (4-bit)

(Module File: CLA_4 .v, Test: CLA_4_TestBench.v)

Circuit Diagram:




A **carry look-ahead adder** improves speed by reducing the amount of time required to determine carry

bits. It calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger-value bits of the adder (in **contrast to** the **ripple carry adder** which first calculates sum and carry and subsequent sum stages must wait until carry is calculated).

The logic is as follows:

$$G[i] = A[i] \cdot B[i], \qquad 0 \leq i \leq 3$$

$$P[i] = A[i] \oplus B[i], \qquad 0 \leq i \leq 3$$

Let $C[0] = c_{in}$ then:

$$sum[i] = P[i] \oplus C[i], \qquad 0 \leq i \leq 3$$

$$C[i] = G[i-1] + (P[i-1] \cdot C[i-1]), \qquad 1 \leq i \leq 4$$

Recursively expanding, we get:

$$C[1] = G[0] + (P[0] \cdot C[0]) = G[0] + (P[0] \cdot c_{in})$$

$$C[2] = G[1] + (P[1] \cdot C[1]) = G[1] + (P[1] \cdot G[0]) + (P[1] \cdot P[0] \cdot c_{in})$$

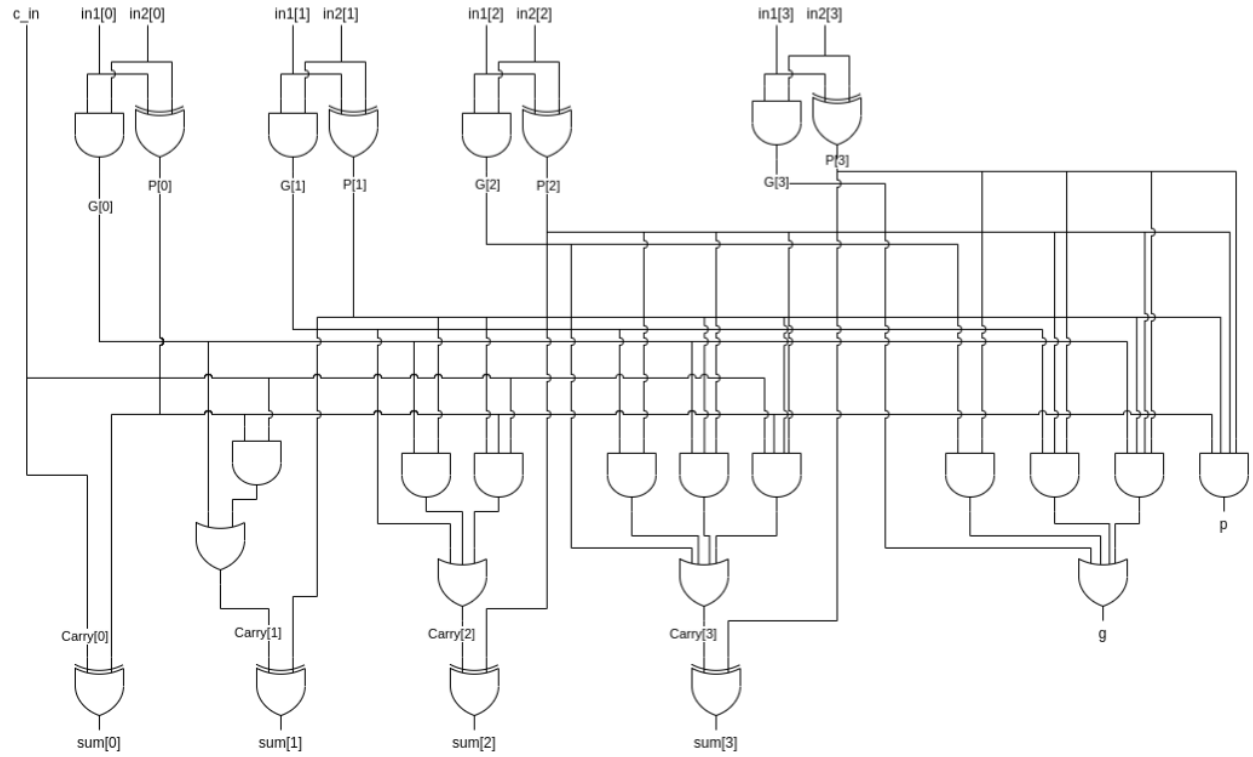
$$\begin{aligned} C[3] &= G[2] + (P[2] \cdot C[2]) \\ &= G[2] + (P[2] \cdot G[1]) + (P[2] \cdot P[1] \cdot G[0]) + (P[2] \cdot P[1] \cdot P[0] \\ &\quad \cdot c_{in}) \end{aligned}$$

$$\begin{aligned} C[4] &= G[3] + (P[3] \cdot C[3]) \\ &= G[3] + (P[3] \cdot G[2]) + (P[3] \cdot P[2] \cdot G[1]) \\ &\quad + (P[3] \cdot P[2] \cdot P[1] \cdot G[0]) + (P[3] \cdot P[2] \cdot P[1] \cdot P[0] \cdot c_{in}) \end{aligned}$$

# CARRY LOOK-AHEAD ADDER (4-bit, augmented)

(Module File: CLA_4_augmented.v, Test: CLA_4_augmented_TestBench.v)

Circuit Diagram:




In this case instead of generating the carry out ,i.e., carry[4] we give the block propagate and generate as output which are then used by the carry lookahead unit. This leads to a modular design using which we can make 16, 32 and 64 bit adders by combining the block propagate and generate from the lower levels instead of rippling the carry out every time. The other logic remains the same as the normal 4-bit CLA.

The block propagate and generates are calculated as follows:

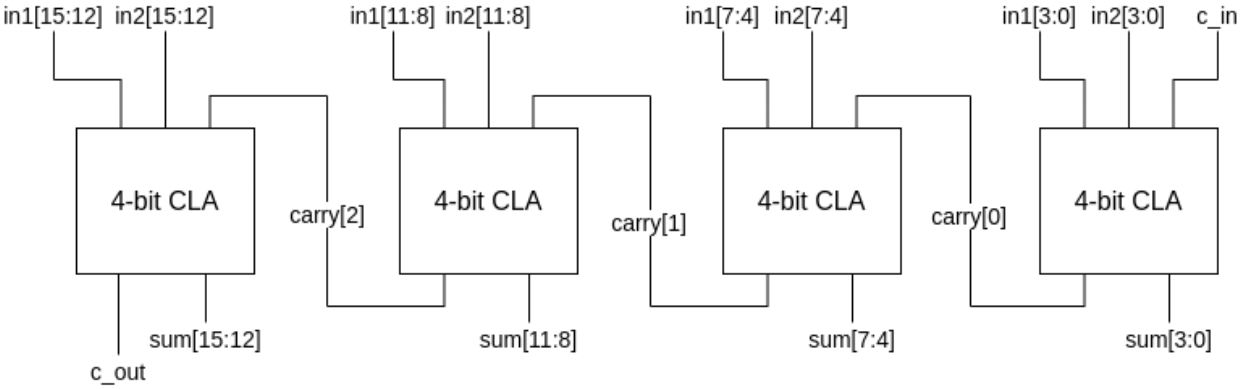
$$p = P[3] \cdot P[2] \cdot P[1] \cdot P[0]$$

$$g = G[3] + P[3] \cdot G[2] + P[3] \cdot P[2] \cdot G[1] + P[3] \cdot P[2] \cdot P[1] \cdot G[0]$$

# CARRY LOOK-AHEAD ADDER (16-bit, cascaded)

(Module File: CLA_16.v, Test: CLA_16_TestBench.v)

This 16-bit adder is made by cascading four 4-bit CLAs by rippling the carry out from one block to another as shown in the figure.

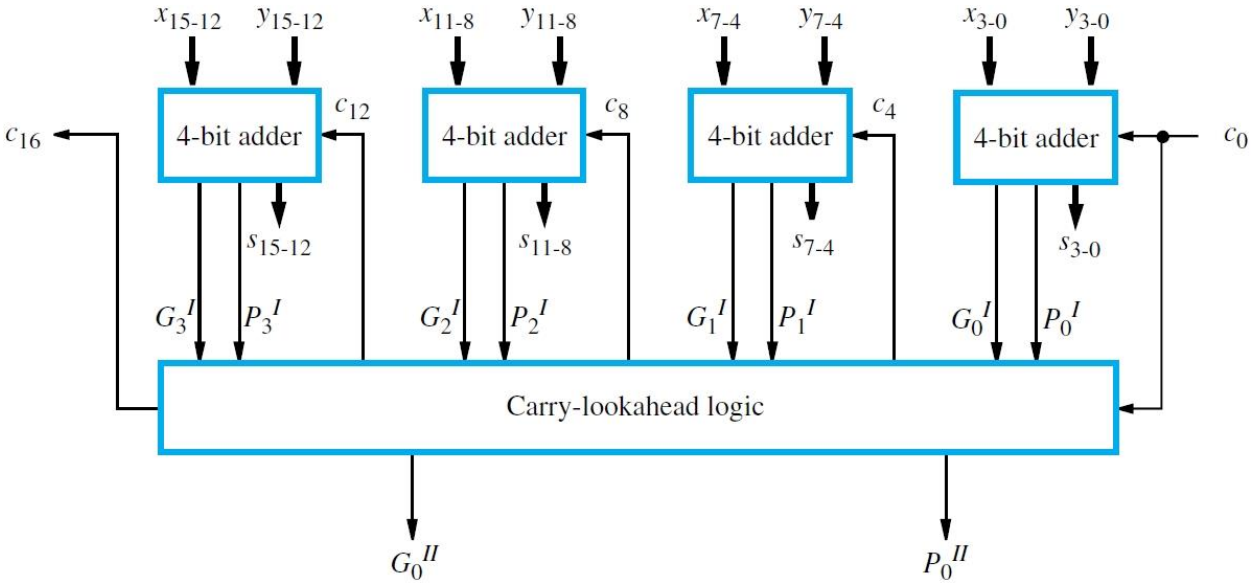



Here, $C[2:0]$ are internal carries which are being propagated (or rippled) forward through the circuit.

# CARRY LOOK-AHEAD ADDER (16-bit, look-ahead)

(Module File: CLA_16_LCA.v, Test: CLA_16_LCA_TestBench.v)

Circuit Diagram:




Instead of rippling the carry bit, we take the carry-lookahead logic to a higher stage. The G and P bits

returned by the 4-bit adder are labelled as $G^I$ and $P^I$. Correspondingly, we extend the same logic which was used to form $G^I, P^I$ to the next stage to get $G^{II}$ and $P^{II}$. This design reduces the delay in the circuit and allows us to use the 16-bit adder in the future for designing higher bit adders in a modular fashion.

The relevant equations are as follows:

Take $c_{in}$ to be $C[0]$ then:

$$C[0] = G^I[i-1] + P^I[i-1] \cdot C[i-1], \qquad 1 \leq i \leq 4$$

Recursively expanding, we get:

$$C[1] = G^I[0] + P^I[0] \cdot C[0] = G^I[0] + P^I[0] \cdot c_{in}$$

$$C[2] = G^I[1] + P^I[1] \cdot C[1] = G^I[1] + P^I[1] \cdot G^I[0] + P^I[1] \cdot P^I[0] \cdot c_{in}$$

$$\begin{aligned} C[3] &= G^I[2] + P^I[2] \cdot C[2] \\ &= G^I[2] + P^I[2] \cdot G^I[1] + P^I[2] \cdot P^I[1] \cdot G^I[0] + P^I[2] \cdot P^I[1] \\ &\quad \cdot P^I[0] \cdot c_{in} \end{aligned}$$

$$\begin{aligned} C[4] &= G^I[3] + P^I[3] \cdot C[3] \\ &= G^I[3] + P^I[3] \cdot G^I[2] + P^I[3] \cdot P^I[2] \cdot G^I[1] + P^I[3] \cdot P^I[2] \\ &\quad \cdot P^I[1] \cdot G^I[0] + P^I[3] \cdot P^I[2] \cdot P^I[1] \cdot P^I[0] \cdot c_{in} \end{aligned}$$

Final Propagate and Generate bits are calculated in just the same way as 4-bit CLA:

$$P^{II} = P^I[3] \cdot P^I[2] \cdot P^I[1] \cdot P^I[0]$$

$$G^{II} = G^I[3] + P^I[3] \cdot G^I[2] + P^I[3] \cdot P^I[2] \cdot G^I[1] + P^I[3] \cdot P^I[2] \cdot P^I[1] \cdot G^I[0]$$

# SYNTHESIS SUMMARY & TIME DIFFERENCE

## Comparison between 16-bit Adders

| | 16-bit CLA Adder (ripple carry unit) | 16-bit CLA Adder (look-ahead carry unit) | 16-bit RCA Adder |
|---|---|---|---|
| **Delay (in $ns$)** | 6.369 | 6.277 | 9.407 |
| **Levels of Logic** | 14 | 11 | 36 |
| **Number of Slice LUTs** | 24 | 43 | 32 |

CLA Adder with the additional look-ahead unit uses lesser logic levels, therefore propagating the sum and carry faster.

## Comparison between 4-bit Adders

| | 4-bit RCA | 4-bit CLA | 4-bit CLA (augmented) |
|---|---|---|---|
| **Delay (in $ns$)** | 3.016 | 2.571 | 2.564 |
| **Levels of Logic** | 10 | 3 | 3 |
| **Number of Slice LUTs** | 8 | 6 | 9 |

A CLA uses much lesser levels of logic as compared to an RCA which has a delay of $2 \cdot n$ for adding two $n$-bit numbers. It can be observed that this additional delay contributes to a speed up of $3.5$x