Simulation of N-bit CLA architecture using 4-bit CLA blocks

Devesh Bhaskaran, Chiranjeevi G N

*Dept. of Electronics and Communication   
PES University, EC Campus   
Banglore 560100*

*Abstract*—In this report we present the simulation of N-bit CLA architecture using a standard 4-bit CLA block. This architecture has been modelled in ModelSim Altera tool using Quartus Prime. The design cycle comprises the schematics, layout generation and verification with arbitrary numbers.

Keywords—CLA, PG-adder

# Introduction

# In the past few years some work on adders mainly for regular structures has been reported, e.g., for multipliers and dividers. Since these blocks are excessively used for diverse applications, the additional effort in constructing a flexible and parameterizable adder seems to be justifiable.

# The advantages of standard block approach are the well-developed techniques of reusability, space efficiency, and defined routing. The disadvantages are that the designer must take care of the block types style of coding for uniformity and find out an appropriate representation. This makes verification and testing, rather simpler.

## CLA Adder

The carry look-ahead adder expresses, carry input at any block is independent of the carry bits generated at the independent blocks. Here the output of any stage is dependent only on the bits which are added in the previous blocks and the carry input provided at the beginning stage. Hence, the circuit at any block does not have the same delay as a conventional ripple carry adder would contain and carry bit of each block can be evaluated at any instant of time.

## CLA Principles

For the addition of two n-bit binary numbers A[2N -1:0] and B[2N -1:0], the well-known PG approach is used :

P(i) = A(i) Ꚛ B(i) (1)

G(i) = A(i) . B(i) (2)

With G(i) and P(i) variables, the carry signal for each bit can be constructed in a recursive way as :

Ci = G i:0 + P i:0 C i-1 (3)

S = P Ꚛ C in (4)

For each 4-bit block the equations which govern the group propagate/generate are :

P3:0 = P3 P2 P1 P0 (5)

G3:0 = G3 + P3 (G2 + P2 (G1+ P1 G0)) (6)

Cout = G 3:0 + P 3:0 C in (7)

Hence for a N-bit architecture the number of standard blocks vary and are used consecutively to yield the addition.

# CLA Adder Logic Design

CLA adder architecture under consideration consists of a hierarchical style of modelling i.e., smaller blocks of the code are framed and recursively called to compute a larger size architecture. In this report, we consider a N-bit architecture approach which gives the user the flexibility to change the architecture as per requirements. For our test case purpose, we fix the value of N to be 3, 4, 5, and 6 and test the results accordingly.

This style of CLA architecture is valid only for bit size which is the integral multiples of 4 as the smallest block consists of 4-bit CLA architecture. We usually find CLA architectures to be efficient above 16 bits of size as the computational speed is better than a ripple carry adder from 16 bits and more.

The primitive block used in this architecture modelling is a 4-bit CLA block and is coded in dataflow style of modelling which is represented below :

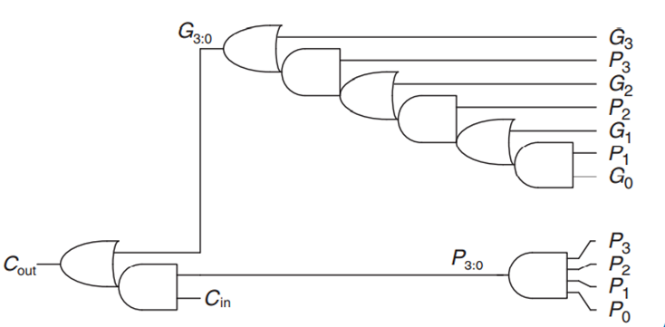


Figure 1: Dataflow model of 4-bit CLA architecture

Figure 1 represents a dataflow model which uses the concept of PG adder or the propagate/generate adder which is a representation of a half adder where the sum acts as the propagate signal and the carry as the generate signal.

These data lines are further used to compute the sum and carry of each block with the equations (4), (5), (6) and (7).

## Architecture Modelling

The following model is a representation for an architecture ranging from N equals 3 to 6 as further increasing the value of N results in a design failure which is due to the lack of accommodation of I/O ports in the Intel MAX DE10 FPGA. The lower limit of N being 3 is also due to the fact that a lower value of N will result in the architecture imbalance as the smallest unit of the model is a 4-bit CLA block.

We use the concept of generate in System Verilog to perform recursive block calling and hence the architecture is flexible but fixed to device requirements. The test bench of such a code must be written for specific size of architecture.

## Dataflow

There are three parts of the code which consists of the basic block i.e., 4-bit CLA block , N-bit CLA architecture and the test bench code to verify the architecture.

Figure 2: Flow chart of Carry Look Ahead Adder

Figure 2 represents the schematic of the CLA architecture where the inputs are first converted to ith propagate and generate signals. Further, we convert these signals to group propagate and generate signals for generation of carry out for a block. Finally we compute the sum by considering the propagate and input carry of the block to produce the output of the block.

module CLA\_N#(parameter N=2)( input logic[2\*\*N-1:0]A, B,

input logic Cin,

output logic[2\*\*N-1:0]S,

output logic Cout );

The above code represents the outline for the N-bit architecture. This helps to realise the module efficiently as the value of parameter can be changed as per requirements which in turn gets reflected in the entire architecture.

## Adder Configurations

From the code we can realize four architectures namely 8, 16, 32, and 64 bits respectively.

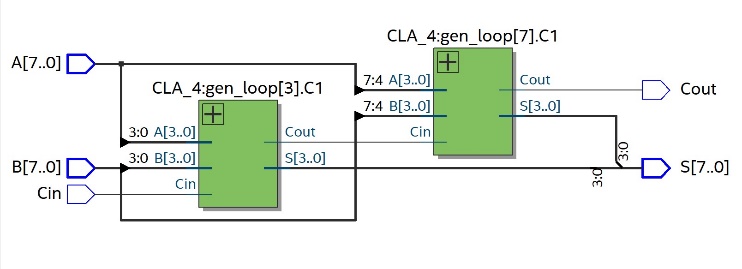


Figure 3.1: 8-bit CLA architecture

Figure 3.1 represents a 8-bit CLA architecture which consists of two 4-bit CLA blocks. Here the carry is propagated only once between the blocks. First four bits is given as input to the first block and the output of the block gives the first four bits of sum and the carry out signal which is fed as input to the second block with the next four bits fed as input signals to give the final carry out signal and the rest four bits of sum.

The same format is followed for the other architectures with the carry of the current being passed as the carry in to the successive block.

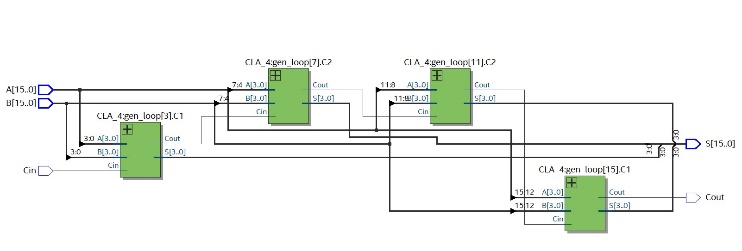


Figure 3.2: 16-bit CLA architecture

Figure 3.2 represents a 16-bit CLA architecture which consists of four 4-bit CLA blocks. Here the carry is propagated thrice between the blocks. As discussed earlier the full input is broken as a token of 4-bits and fed to each block to give the final output while propagating the carry out from each block to the input carry of the next block.

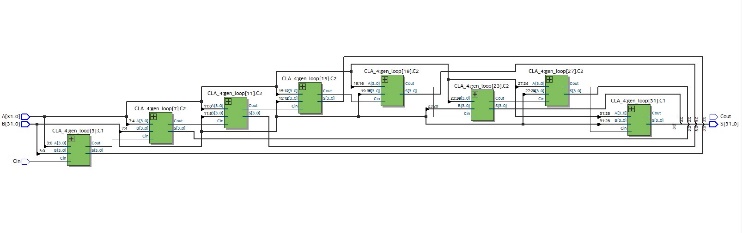


Figure 3.3: 32-bit CLA architecture

Figure 3.3 represents a 32-bit CLA architecture which consists of eight 4-bit CLA blocks. Here the carry is propagated seven times between the blocks.

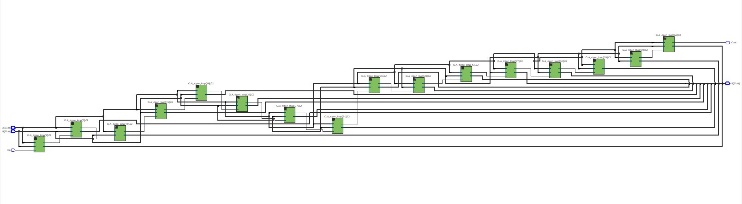


Figure 3.4: 64-bit CLA architecture

Figure 3.3 represents a 64-bit CLA architecture which consists of sixteen 4-bit CLA blocks. Here the carry is propagated fifteen times between the blocks.

The sum of two inputs is consecutively added by computing 4-bits of data at each iteration. The process occurs parallelly in sync with the carry generation at the end of each iteration. Hence at the end of computation both the carry out and sum of the 2N bits input are computed.

# Results

The delay equation for a carry look-ahead adder is represented by the equation (8). This equation is applicable for real time simulation as there are delays in each gate propagation but for technical purpose we do not consider in our model.

tCLA = tpg +tpg\_block + ((N/k)−1)\*tAND\_OR +k\*tFA (8)

Simulation of variable architecture consisting of N valued from 3 to 6 on the ModelSim Altera platform produces the following results :

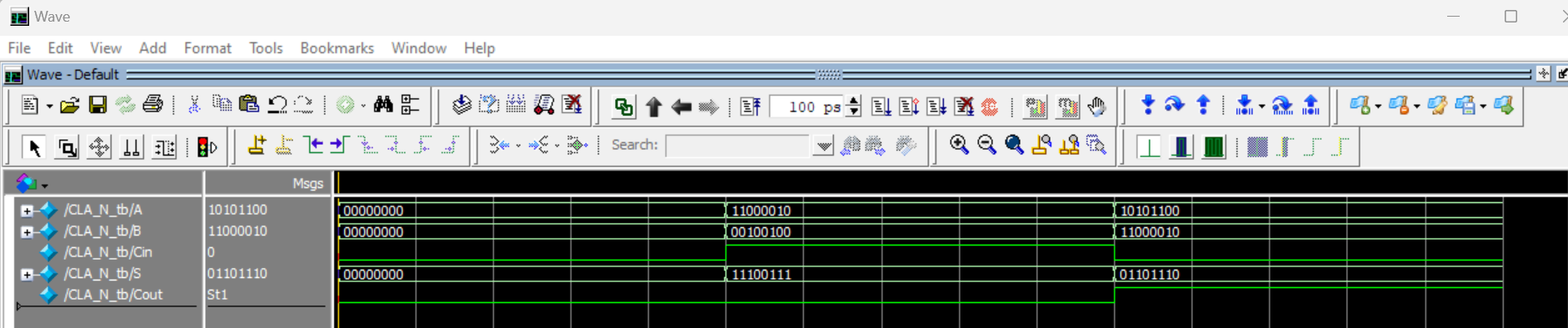


Figure 4.0: Simulation of an 8-bit CLA

Figure 4.0 represents the simulation of a 8-bit carry look ahead adder with the initial conditions to be forced to 0. When the value of the inputs are fed as

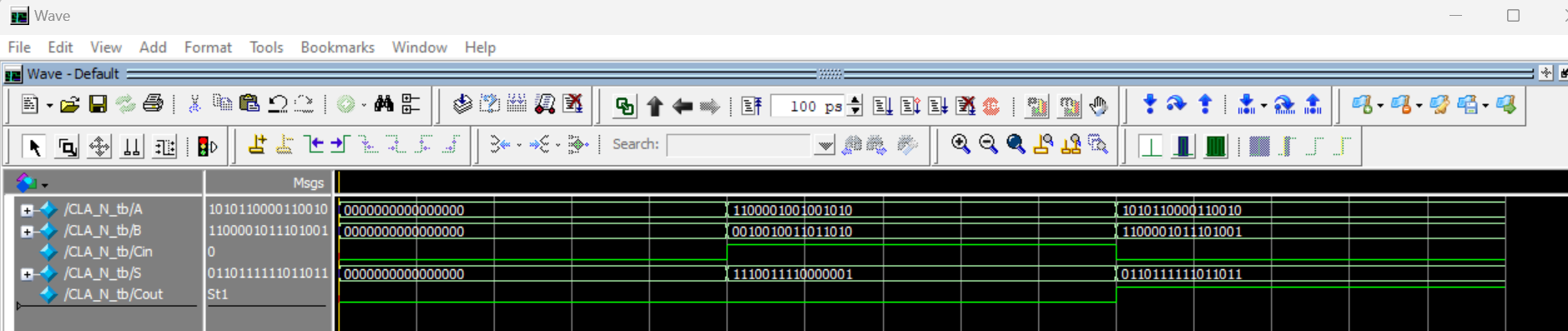


Figure 4.1: Simulation of a 16-bit CLA

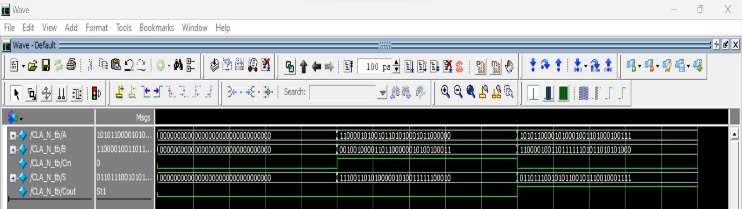


Figure 4.2: Simulation of a 32-bit CLA

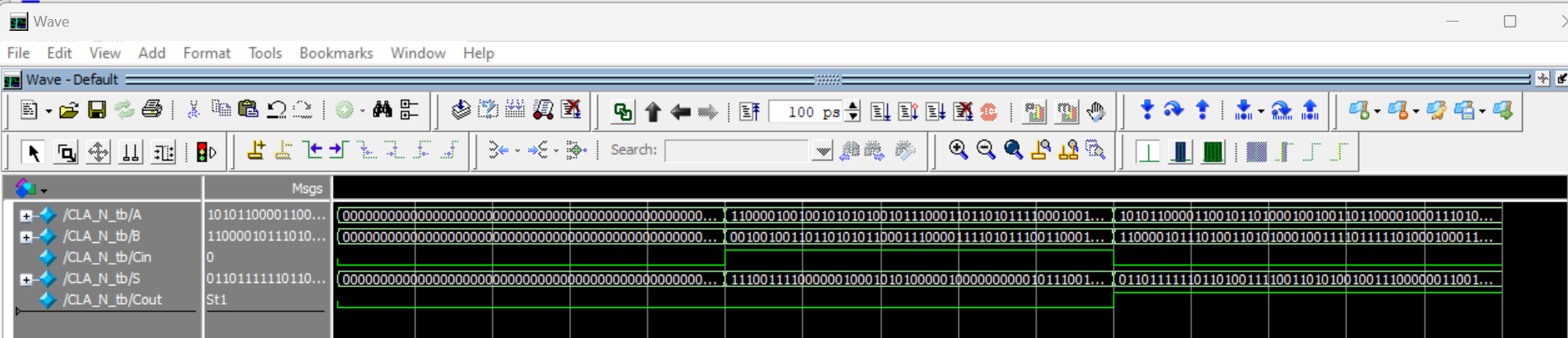


Figure 4.2: Simulation of a 64-bit CLA

The utilization report of the 4-bit CLA block and the final architecture when the value of N is 3,4,5 and 6 is shown below:

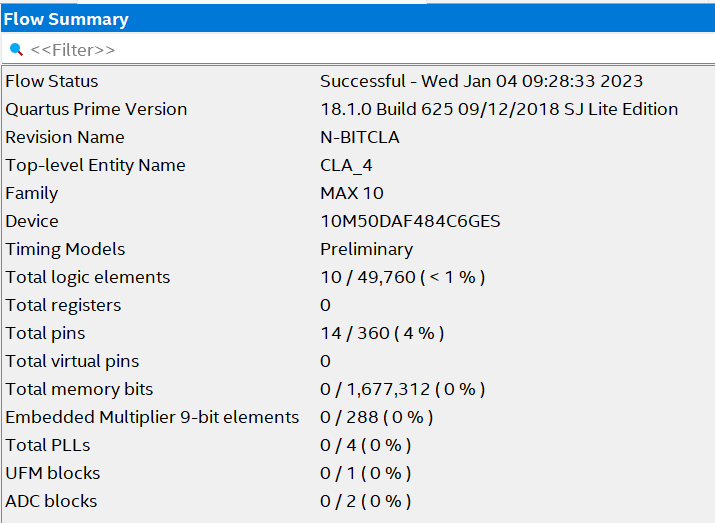


Figure 5.0: Utilization report of 4-bit CLA block

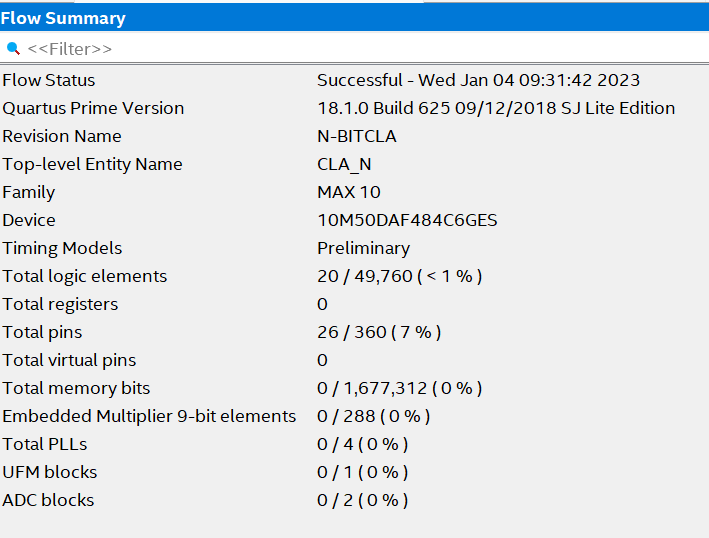


Figure 5.1: Utilization report of a 8-bit CLA architecture

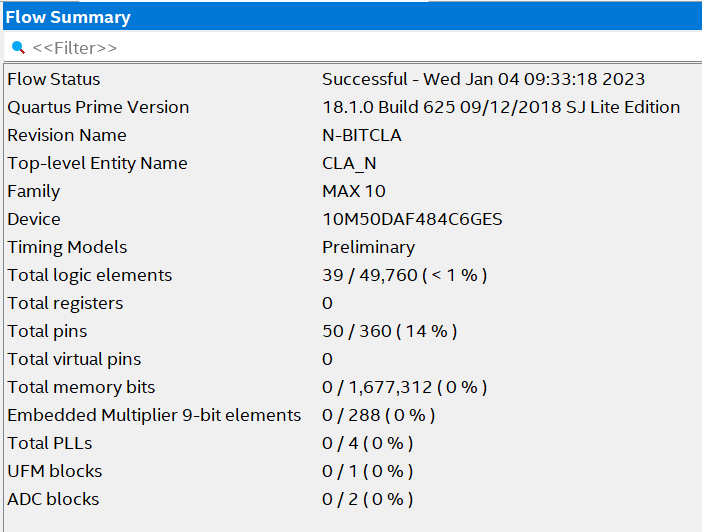


Figure 5.2: Utilization report of a 16-bit CLA architecture

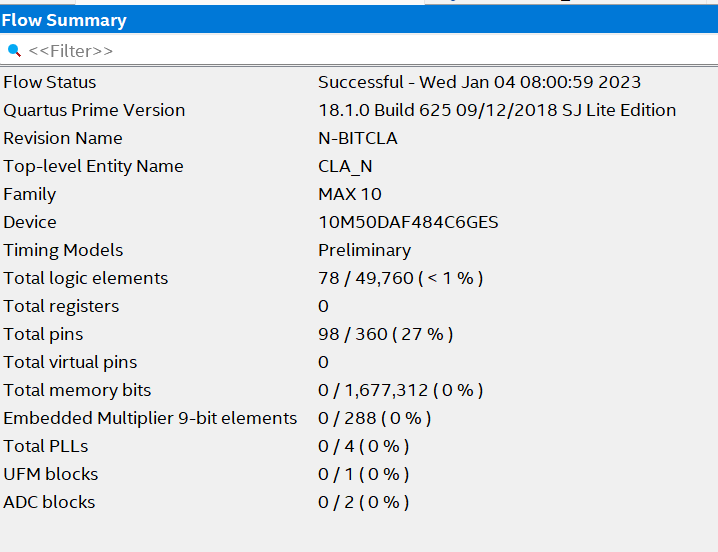


Figure 5.3: Utilization report of a 32-bit CLA architecture

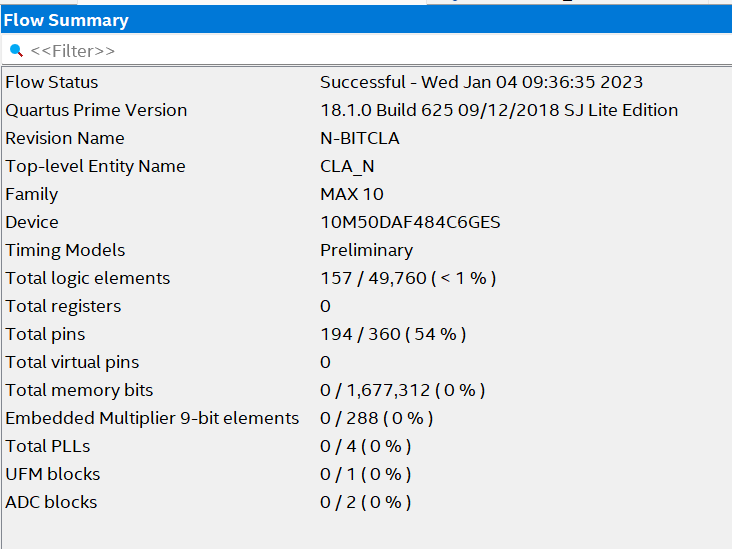


Figure 5.4: Utilization report of a 64-bit CLA architecture

# Observations

The following observations were noted during the compilation of the code:

1. The architecture represents the CLA adder for values of N ranging from 3 to 6. This gives the flexibility to change the architecture according to the system requirements.
2. Any value lower than 3 results the architecture to reach its lowest block which is a 4-bit CLA architecture.
3. N value exceeding 6 will lead to design failure as the FPGA used for testing doesn’t contain more than 360 pins and hence the code will not be synthesizable.
4. Future implementation of this architecture would be to change the lowest block to a flexible architecture from 4-bit to a M-bit CLA adder.

# Conclusions

Carry Look-ahead Adders are implemented as ALU’s in IC’s. Hence, it is easy to embed the adder in circuits. Here the increase in the number of gates is also moderate when used for higher bits.The designs are simple so as to not require a large number of computational resources.

However, each implementation accurately reflects the hierarchical mode of data computation. Hence the design can be expanded for a wide range of architectures and also further used in and ALU design.

The future possibilities of this architecture are that it can be made flexible at the lowest block which would be a 2M block rather than a conventional 4-bit block.

##### References

1. David Jeff Jackson and Sidney Joel Hannah, “Modelling and Comparison of Adder Designs with Verilog HDL,” 0094-2898/93 $03.00 0 1993 IEEE
2. David Money Harris, Sarah L. Harris, “Digital Design and Computer Architecture,”Second edition , pp. 241–243.