ACA Assignment 1
64-bit Carry Look Ahead Adder

Group 2

AU2140039 Nish Apoorva Parikh

AU2140206 Rutul Niralkumar Patel

AU2140213 Vanaja Binay Agarwal

64 - BIT CARRY LOOK AHEAD ADDER

➢ The 64-bit Carry Lookahead Adder (CLA) is an advanced and efficient method to add two 64-bit binary numbers, and it significantly improves the addition time when compared to the traditional ripple-carry adder.

➢ 16-bit CLA Modules: The core building block is the 16-bit CLA module, which calculates carries and sum bits for a 16-bit addition.

- P (Carry Propagator): P signals if a carry will spread from a bit to the next through XOR of operand bits.
- G (Carry Generator): G signals if a carry is generated at a bit position via AND of operand bits, regardless of incoming carry.

➢ 4-bit CLA Modules: These are even smaller building blocks that handle 4 bits at a time. They serve as the fundamental units for constructing the 16-bit modules.

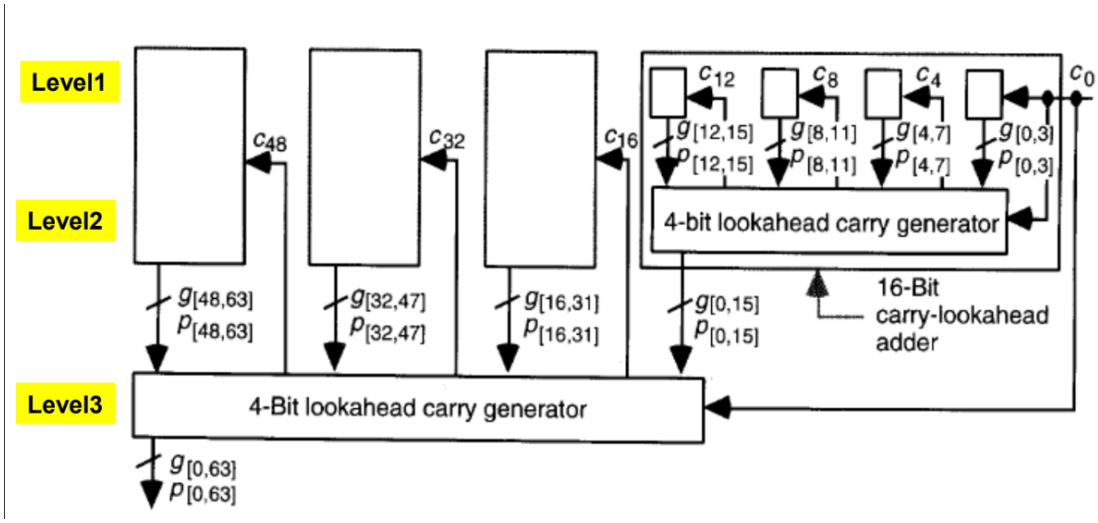
➢ 64-bit Composition: To create a 64-bit CLA, you combine four 16-bit CLA modules. Each of these modules manages 16 bits of the input operands.

➢ Carry Propagation: While the 16-bit modules compute most of the carries and sum bits, a final 4-bit CLA module determines how carries propagate between the 16-bit modules.

➢ Parallel Sum Calculation: After carry determination, the sum bits for the 64-bit addition are computed in parallel, taking advantage of known carries

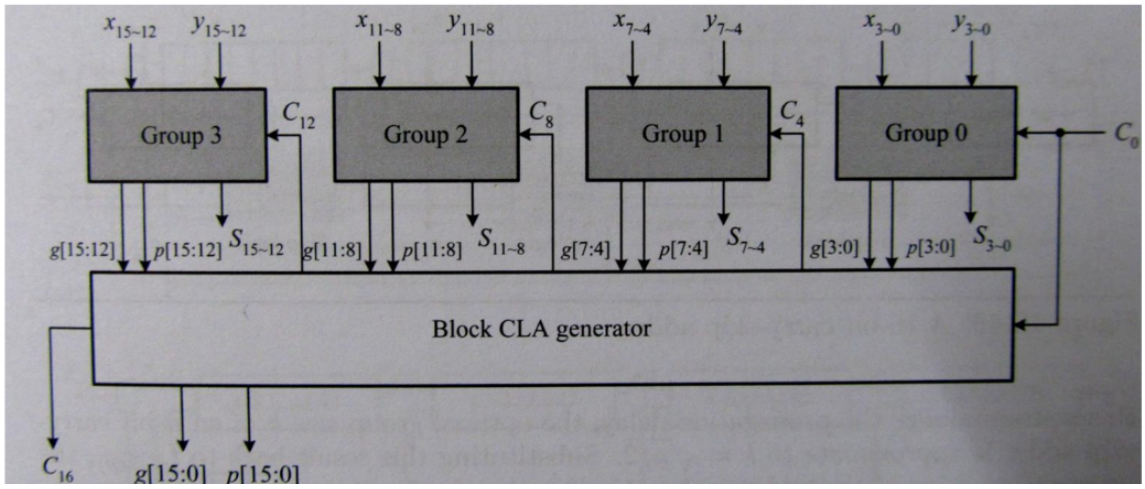
MODULES

CARRY LOOK AHEAD 64 BIT ADDER

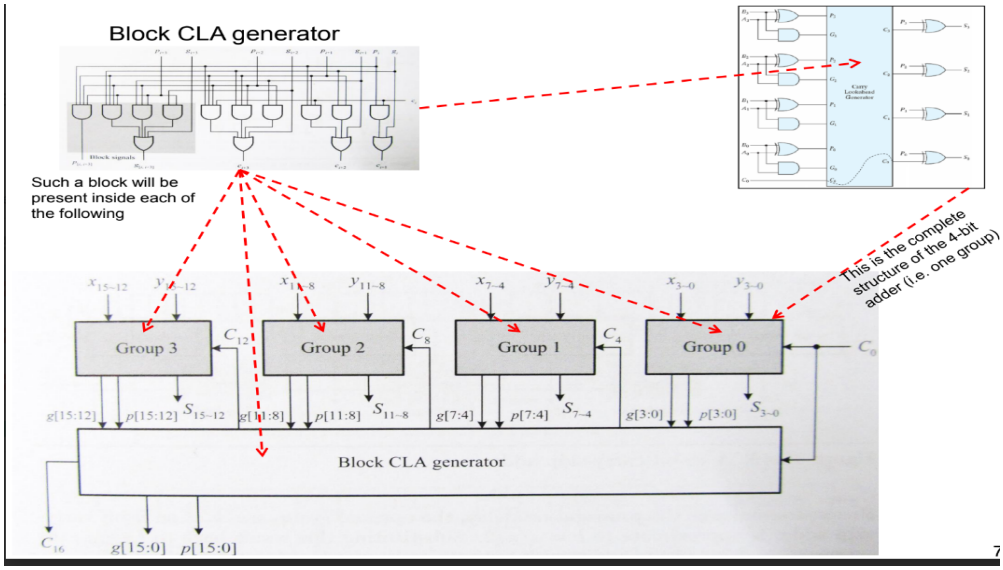



```
299  module cla_64_bit(A, B, Cin, S, Carry, Cout);
300
301  input [63:0] A, B;
302  input Cin;
303  output wire [63:0] S;
304  output wire [63:0] Carry;
305  output wire Cout;
306
307  wire carry1, carry2, carry3;
308  wire [3:0] g, p;
309  wire c1, c2, c3;
310  cla_16_bit inst0(A[15:0], B[15:0], Cin, S[15:0], Carry[15:0], carry1, g[0], p[0]);
311  cla_16_bit inst1(A[31:16], B[31:16], c1, S[31:16], Carry[31:16], carry2, g[1], p[1]);
312  cla_16_bit inst2(A[47:32], B[47:32], c2, S[47:32], Carry[47:32], carry3, g[2], p[2]);
313  cla_16_bit inst3(A[63:48], B[63:48], c3, S[63:48], Carry[63:48], Cout, g[3], p[3]);
314  block_cla_generator inst4(Cin,g[0], g[1], g[2], g[3], p[0], p[1], p[2], p[3], G, P,c1,c2,c3);
315
316
317  endmodule
318
```

CARRY LOOK AHEAD 16 BIT ADDER

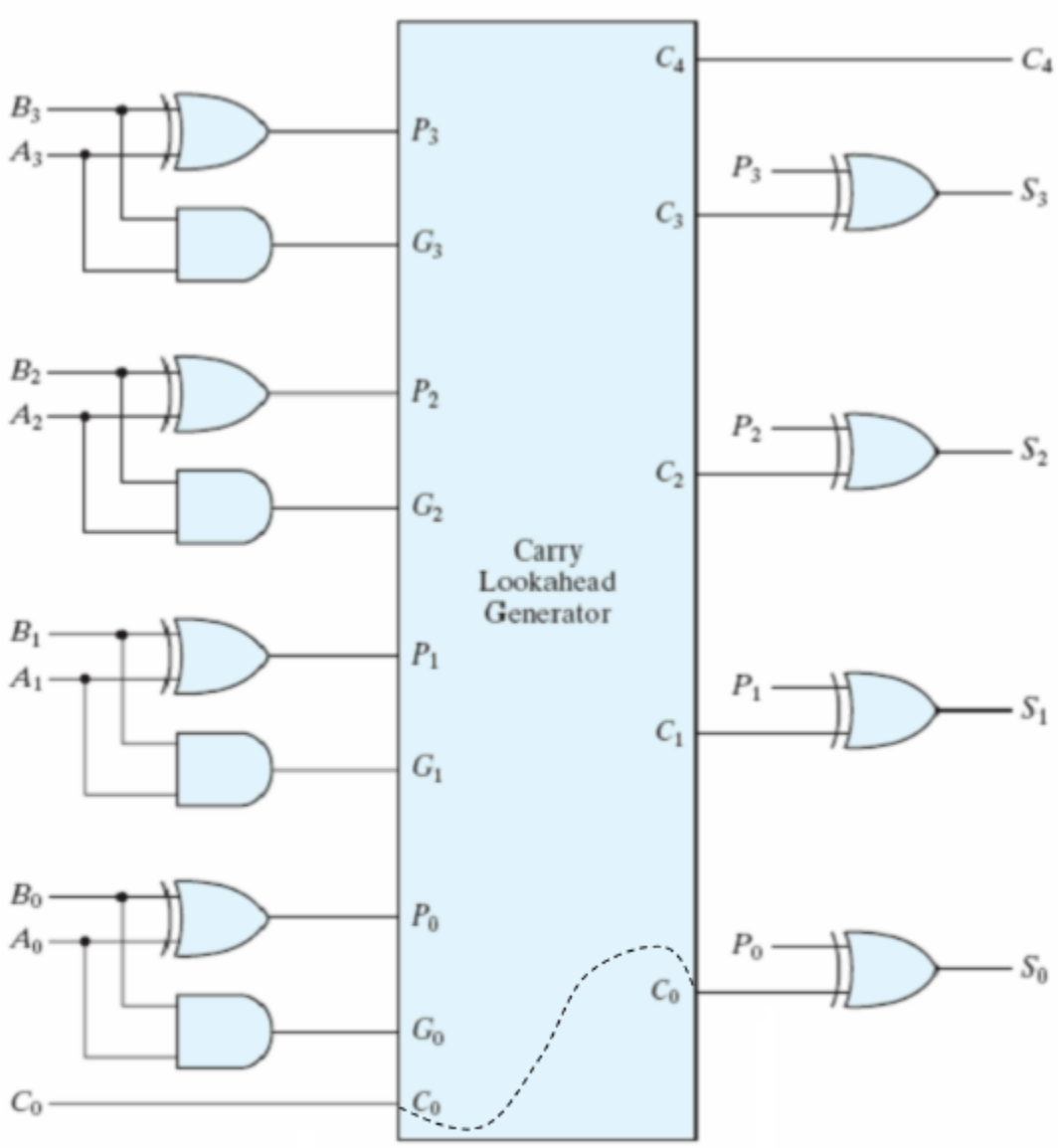



```
module cla_16_bit(A,B,Cin,S,Carry, Cout,G,P);

input [15:0] A, B;
input Cin;
output wire [15:0] S;
output wire [15:0] Carry;
output wire G, P, Cout;
wire c1, c2, c3;
wire carry1, carry2, carry3;
wire [3:0] g, p;

cla_4_bit inst0(A[3:0], B[3:0], Cin, S[3:0],Carry[3:0], carry1, g[0], p[0]);
cla_4_bit inst1(A[7:4], B[7:4], c1, S[7:4],Carry[7:4], carry2, g[1], p[1]);
cla_4_bit inst2(A[11:8], B[11:8], c2, S[11:8], Carry[11:8], carry3, g[2], p[2]);
cla_4_bit inst3(A[15:12], B[15:12], c3, S[15:12], Carry[15:12], Cout, g[3], p[3]);
block_cla_generator inst4(Cin,g[0], g[1], g[2], g[3], p[0], p[1], p[2], p[3], G, P,c1,c2,c3);

endmodule
```




```
354  module block_cla_generator(Cin,g0,g4,g8,g12,p0,p4,p8,p12,G,P,c4,c8,c12);
355
356  input g0,g4,g8,g12,p0,p4,p8,p12,Cin;
357  output wire G,P,c4,c8,c12;
358
359  myAND4 inst0(P,p0,p4,p8,p12);
360
361  wire t1,t2,t3;
362  myAND4 inst1(t1,p4,p8,p12,g0);
363  myAND3 inst2(t2,p8,p12,g4);
364  myAND2 inst3(t3,p12,g8);
365
366  myOR4 inst4(G,t1,t2,t3,g12);
367
368
369  // c4 = g0 or Cin*p0
370
371  wire tempc4;
372  myOR2 inst5(c4,tempc4,g0);
373  myAND2 inst6(tempc4,Cin,p0);
374
375  // c8 = g4 or g0*p4 or Cin*p0*p4
376
377  wire [1:0] tempc8;
378  myOR3 inst7(c8,tempc8[0],tempc8[1],g4);
379  myAND2 inst8(tempc8[0],g0,p4);
380  myAND3 inst9(tempc8[1],Cin,p0,p4);
381
382  // c12 = g8 or g4*p8 or g0*p4*p8 or Cin*p0*p4*p8
383
384  wire [2:0] tempc12;
385  myOR4  inst10(c12,g8,tempc12[2],tempc12[1],tempc12[0]);
386  myAND4 inst11(tempc12[2],Cin,p0,p4,p8);
387  myAND3 inst12(tempc12[1],g0,p4,p8);
388  myAND2 inst13(tempc12[0],g4,p8);
389
390
391
392  endmodule
393
```

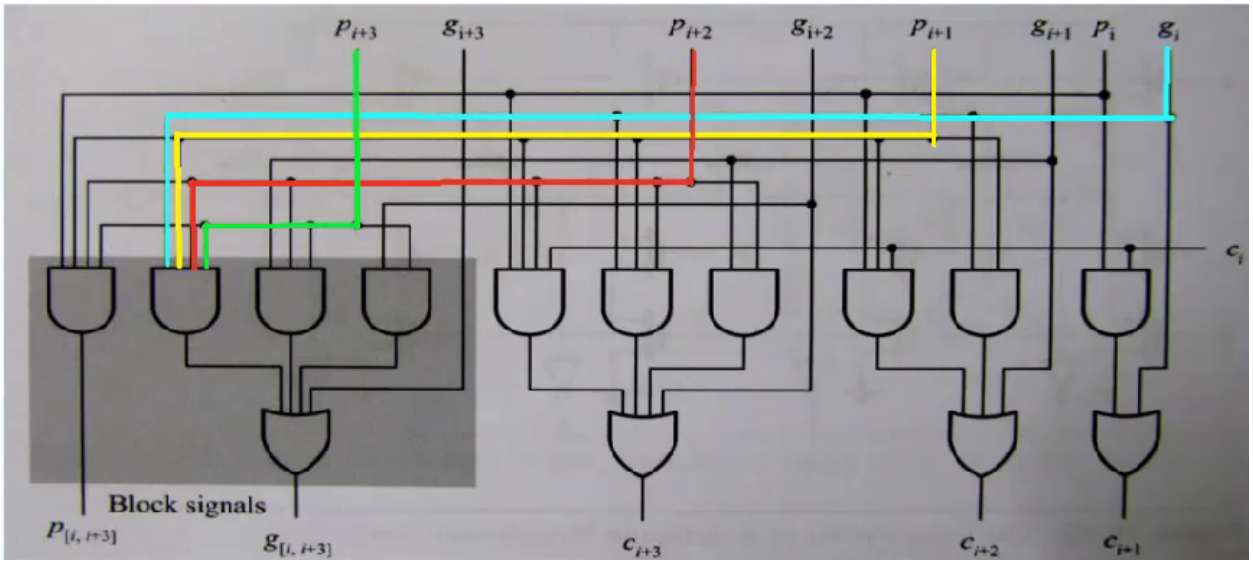
CARRY LOOK AHEAD 4-BIT ADDER



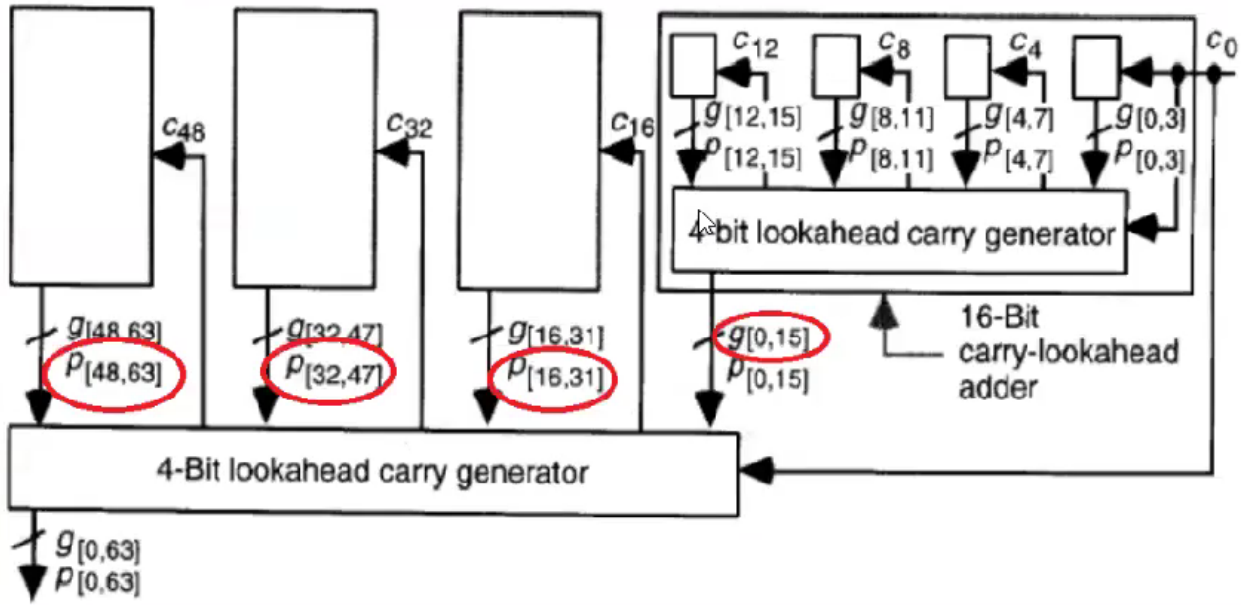
```
402 module cla_4_bit(A,B,Cin,S,Carry,Cout,G,P);
403 input [3:0] A, B;
404 input Cin;
405 output wire Cout, G, P;
406 output wire [3:0] S;
407 output wire [3:0] Carry;
408
409 wire [3:0] g, p;
410 myExOR inst0(p[0],A[0],B[0]);
411 myExOR inst1(p[1],A[1],B[1]);
412 myExOR inst2(p[2],A[2],B[2]);
413 myExOR inst3(p[3],A[3],B[3]);
414
415 myAND2 inst4(g[0],A[0],B[0]);
416 myAND2 inst5(g[1],A[1],B[1]);
417 myAND2 inst6(g[2],A[2],B[2]);
418 myAND2 inst7(g[3],A[3],B[3]);
419
420 myAND4 inst18(P,p[0],p[1],p[2],p[3]);
421
422
423 wire Ctmp0,c1,c2,c3;
424 wire [1:0] Ctmp1;
425 wire [2:0] Ctmp2;
426 wire [3:0] Ctmp3;
427
428 myAND2 inst8(Ctmp0, p[0], Cin);
429
430 myAND2 inst9(Ctmp1[0],p[1],g[0]);
431 myAND3 inst10(Ctmp1[1],p[0],p[1],Cin);
432
433 myAND2 inst11(Ctmp2[0],p[2],g[1]);
434 myAND3 inst12(Ctmp2[1],p[2],p[1],g[0]);
435 myAND4 inst13(Ctmp2[2],p[0],p[1],p[2],Cin);
```

```
437 myAND2 inst14(Ctmp3[0],p[3],g[2]);
438 myAND3 inst15(Ctmp3[1],p[3],p[2],g[1]);
439 myAND4 inst16(Ctmp3[2],p[3],p[2],p[1],g[0]);
440 myAND5 inst17(Ctmp3[3],p[3],p[2],p[1],p[0],Cin);
441
442 myOR2 inst20(c1, Ctmp0, g[0]);
443 myOR3 inst21(c2,Ctmp1[0],Ctmp1[1],g[1]);
444 myOR4 inst22(c3,Ctmp2[0],Ctmp2[1],Ctmp2[2],g[2]);
445 myOR5 inst23(Cout,Ctmp3[0],Ctmp3[1],Ctmp3[2],Ctmp3[3],g[3]);
446
447 //myOR2 inst24(Carry[0], Ctmp0, g[0]);
448 //myOR3 inst25(Carry[1],Ctmp1[0],Ctmp1[1],g[1]);
449 //myOR4 inst26(Carry[2],Ctmp2[0],Ctmp2[1],Ctmp2[2],g[2]);
450 //myOR5 inst27(Carry[3],Ctmp3[0],Ctmp3[1],Ctmp3[2],Ctmp3[3],g[3]);
451 assign Carry[3:0] = {Cout, c3, c2, c1};
452
453 myExOR inst28(S[0],Cin,p[0]);
454 myExOR inst29(S[1],c1,p[1]);
455 myExOR inst30(S[2],c2,p[2]);
456 myExOR inst31(S[3],c3,p[3]);
457
458 myOR4 inst32(G,g[3],Ctmp3[0],Ctmp3[1],Ctmp3[2]);
459
460 endmodule
```

DELAY IN 4-BIT



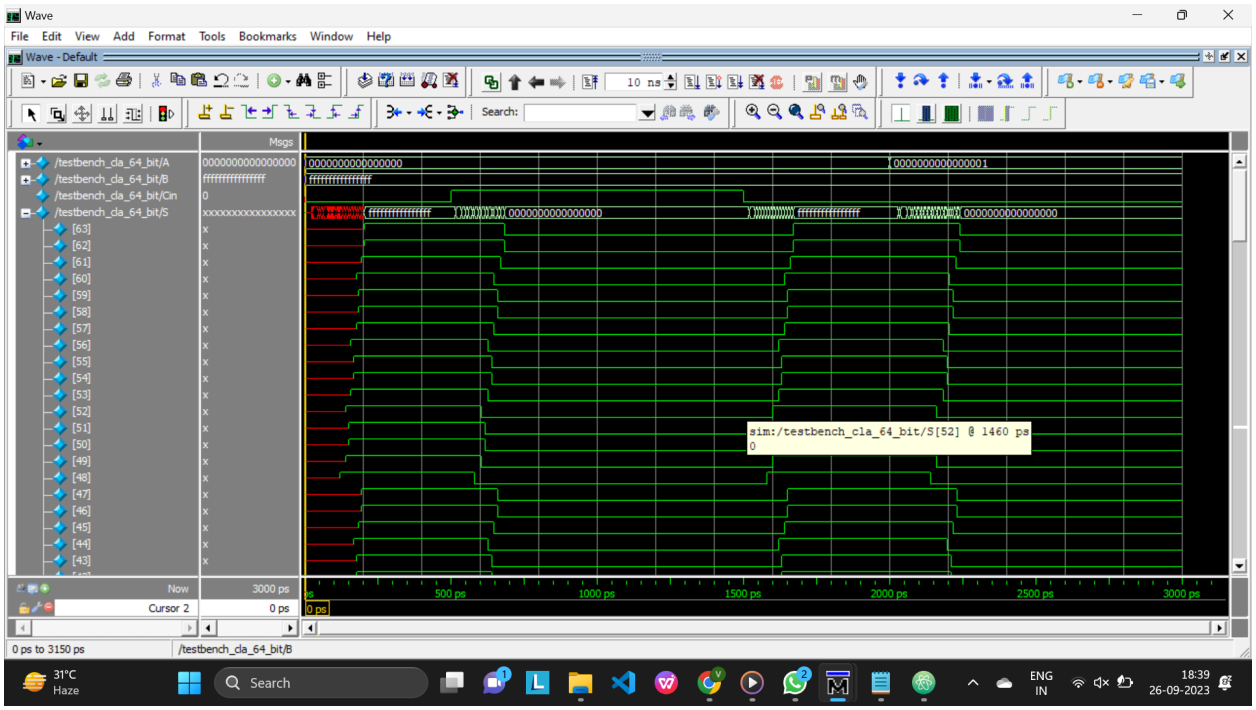
DELAY IN 64-BIT

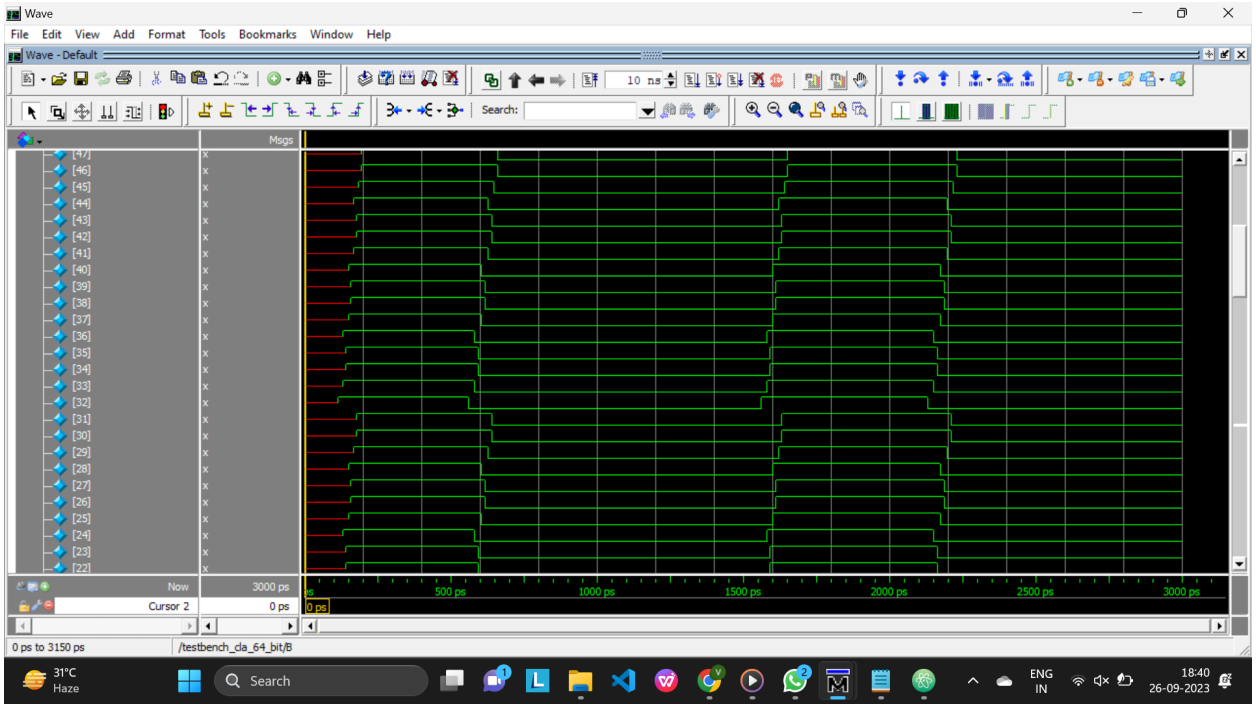


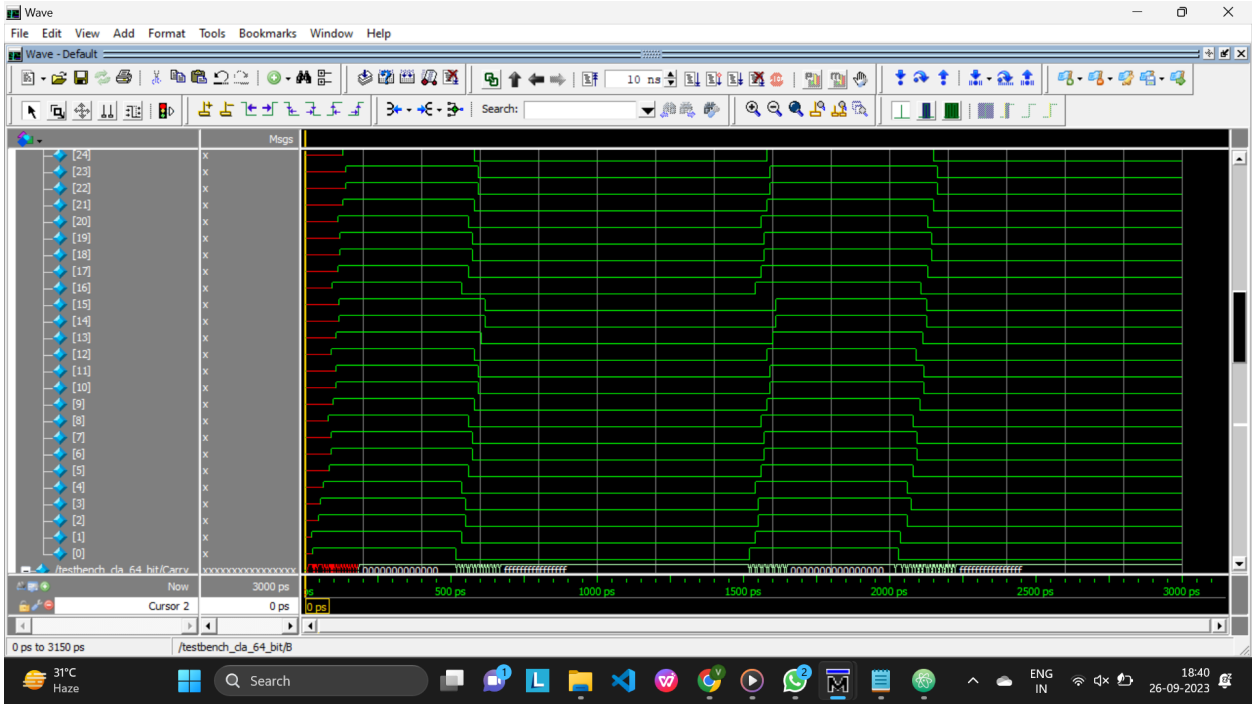
TESTBENCH

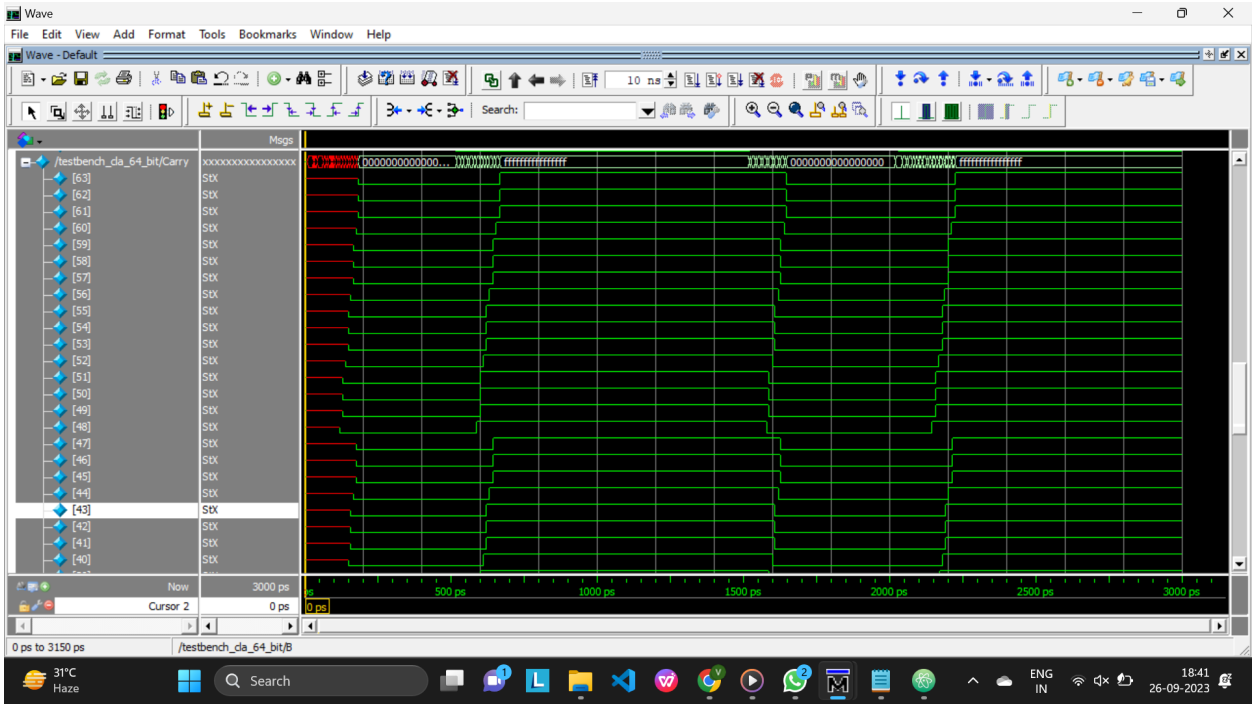
```
463  module testbench_cla_64_bit();
464
465    reg [63:0] A, B;
466    reg Cin;
467    wire [63:0] S;
468    wire [63:0] Carry;
469    wire Cout;
470
471    // Instantiate the CLA 64-bit adder
472    cla_64_bit uut(A,B,Cin,S,Carry,Cout);
473
474    // Clock generation
475    reg clk = 0;
476     always
477       begin
478          #700 clk = ~clk;
479           end
480
481
482    initial
483      begin
484        $dumpfile("wavedump.vcd");
485        $dumpvars();
486        $monitor("Time=%0t A=%b B=%b Cin=%b S=%b Carry=%b Cout=%b", $time, A, B, Cin, S, Carry, Cout);
487        A = 64'b0000000000000000000000000000000000000000000000000000000000000000;
488        B = 64'b1111111111111111111111111111111111111111111111111111111111111111;
489        Cin = 1'b0;
490        #500
491        Cin = 1'b1;
492        #1000
493        A = 64'b0000000000000000000000000000000000000000000000000000000000000000;
494        B = 64'b1111111111111111111111111111111111111111111111111111111111111111;
495        Cin = 1'b0;
496        #500
497        A = 64'b0000000000000000000000000000000000000000000000000000000000000001;
498
499        #1000
500        $finish;
501      end
502
503  endmodule
504
```

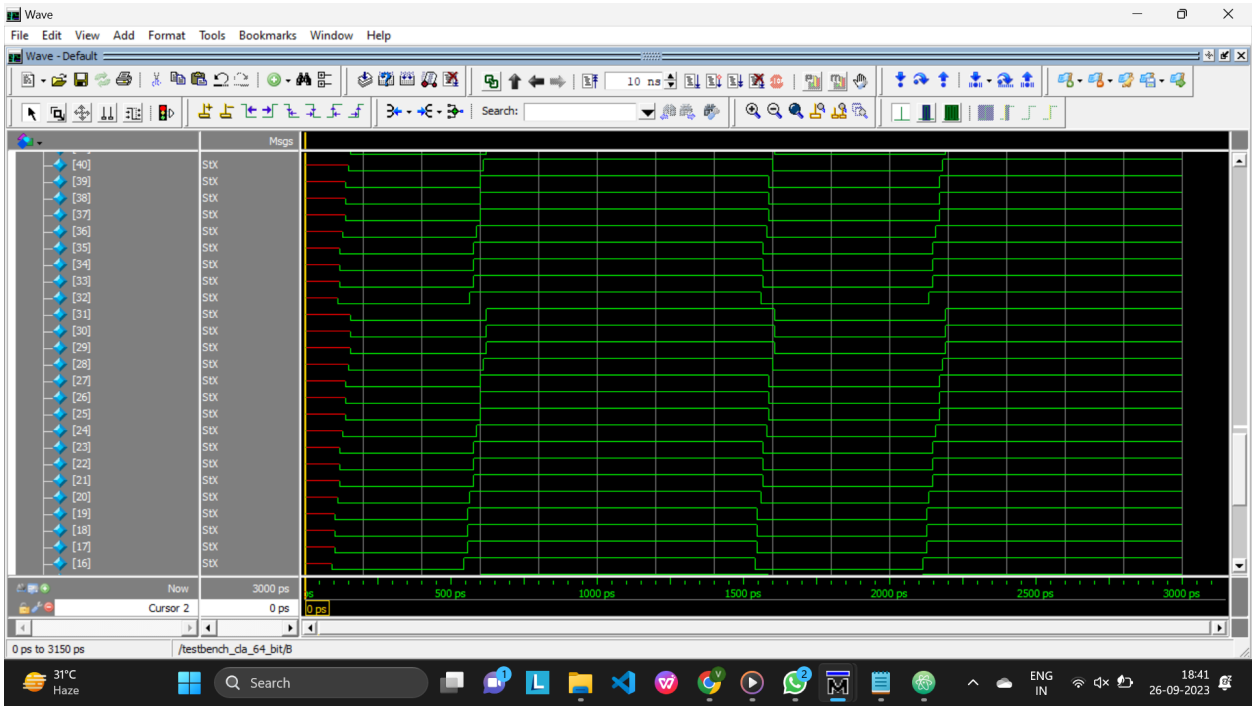
WAVEFORMS

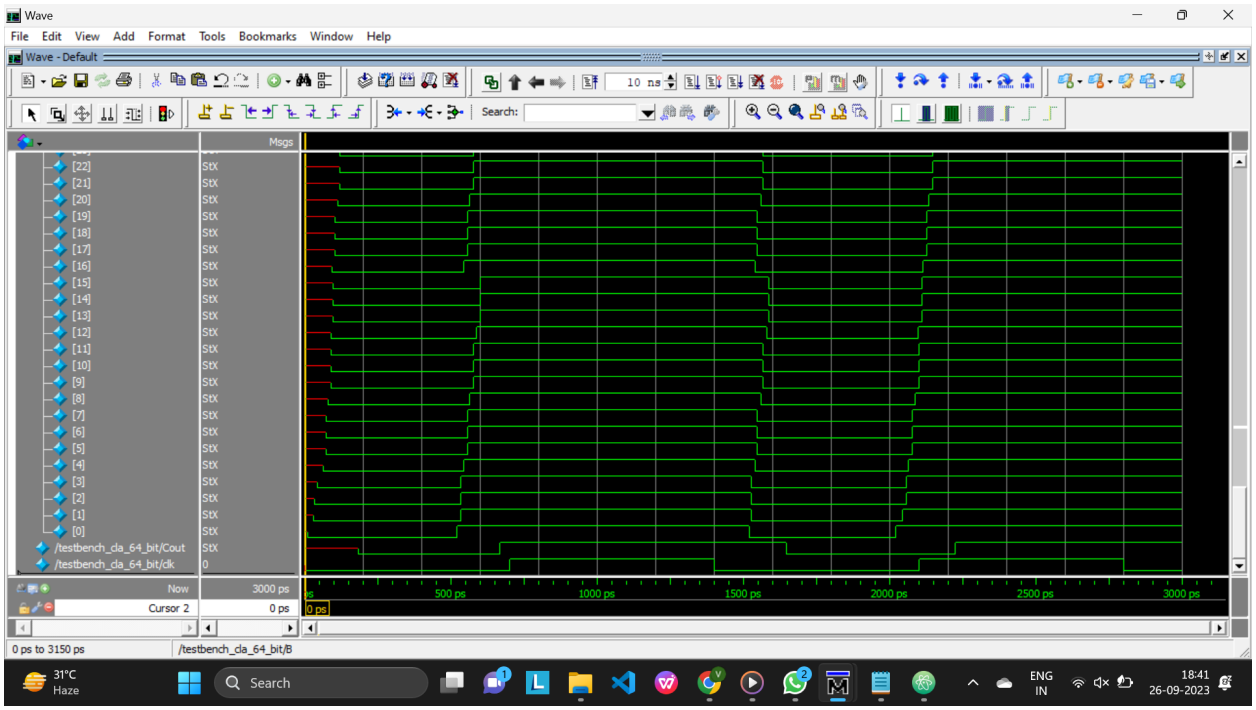

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
/testbench_cla_64_bit/A
/testbench_cla_64_bit/B
/testbench_cla_64_bit/Cin
/testbench_cla_64_bit/S
sim:/testbench_cla_64_bit/S[52] @ 1460 ps
Now
Cursor 2
0 ps to 3150 ps
/testbench_cla_64_bit/B
31°C
Haze
Search
ENG
IN
18:39
26-09-2023


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
Now
3000 ps
Cursor 2
0 ps
500 ps
1000 ps
1500 ps
2000 ps
2500 ps
3000 ps
0 ps to 3150 ps
/testbench_cla_64_bit/B
31°C
Haze
Search
ENG
IN
18:40
26-09-2023

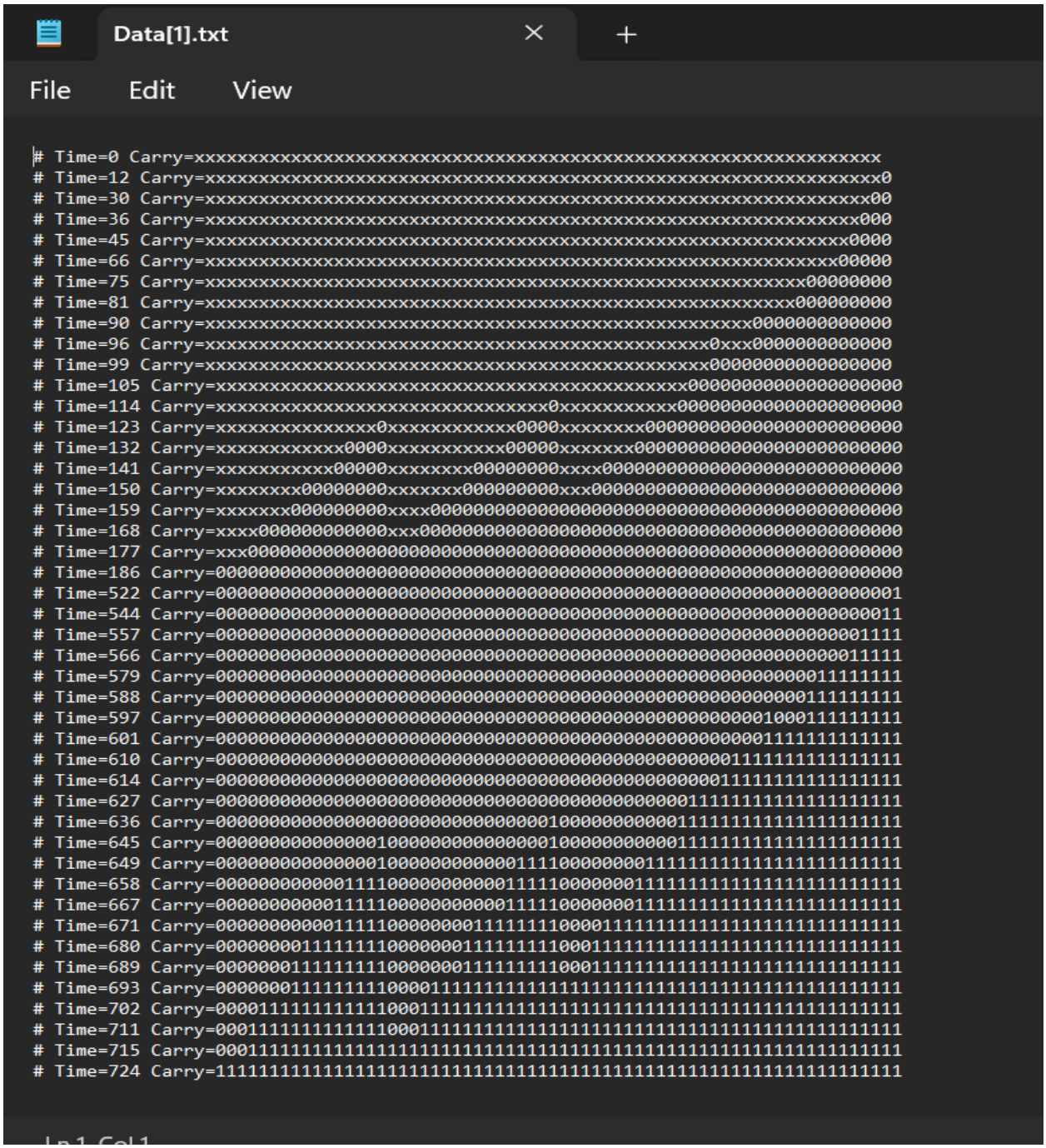

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
Now
3000 ps
Cursor 2
0 ps
500 ps
1000 ps
1500 ps
2000 ps
2500 ps
3000 ps
0 ps to 3150 ps
/testbench_cla_64_bit/B
31°C
Haze
Search
ENG
IN
18:40
26-09-2023


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
/testbench_cla_64_bit/Carry
Msgs
Now
3000 ps
Cursor 2
0 ps
500 ps
1000 ps
1500 ps
2000 ps
2500 ps
3000 ps
0 ps to 3150 ps
/testbench_cla_64_bit/B
31°C
Haze
Search
ENG
IN
18:41
26-09-2023


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
Now
3000 ps
Cursor 2
0 ps
500 ps
1000 ps
1500 ps
2000 ps
2500 ps
3000 ps
0 ps to 3150 ps
/testbench_cla_64_bit/B
31°C
Haze
Search
ENG
IN
18:41
26-09-2023


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
[22] StX
[21] StX
[20] StX
[19] StX
[18] StX
[17] StX
[16] StX
[15] StX
[14] StX
[13] StX
[12] StX
[11] StX
[10] StX
[9] StX
[8] StX
[7] StX
[6] StX
[5] StX
[4] StX
[3] StX
[2] StX
[1] StX
[0] StX
/testbench_cla_64_bit/Cout StX
/testbench_cla_64_bit/clk 0
Now 3000 ps
Cursor 2 0 ps
500 ps 1000 ps 1500 ps 2000 ps 2500 ps 3000 ps
0 ps
0 ps to 3150 ps
/testbench_cla_64_bit/B
31°C Haze
Search
ENG IN
18:41 26-09-2023

CODE USED TO CALCULATE CARRY DELAY



C: > Users > VANAJA AGARWAL > Downloads > Delay_Calculation.py > ...

```python
file = open('Data.txt','r')

data =file.read()


data=data.split(" \n# ")
data = " ".join(data)
data = data.split(" ")
time_list = []
carry_list = []

for item in data:
    if item.startswith('Time='):
        time_list.append(int(item.split('=')[1]))
    elif item.startswith('Carry='):
        carry_list.append(item.split('=')[1])

pointer=0
for i in range(len(time_list)):
    if(time_list[i]>500):
        pointer= i
        break
carry_list = (carry_list[pointer-1::])
time_list = time_list[pointer-1::]
print(len(time_list),len(carry_list))
bin_carry_list = [bin(int(carry, 2))[2:] for carry in carry_list]

Carry_Delay = [0] * 64  # Initialize the Carry_Delay list with zeros

```

```
for i in range(1, len(bin_carry_list)):
    x = int(bin_carry_list[i], 2) - int(bin_carry_list[i - 1], 2)
    print(x)
    place = 0
    while x > 0:
        if x & 1 == 1:
            if place < len(Carry_Delay):  # Check if place is within the bounds
                Carry_Delay[place] = time_list[i] - 500
        x = x >> 1

        place += 1
    print(place)

print((Carry_Delay))

import pandas as pd

df = pd.DataFrame({'Carry_Delay': Carry_Delay})
writer = pd.ExcelWriter('Table_Assignment1.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='Sheet1', startrow=3, startcol=2, index=False)
workbook = writer.book
worksheet = writer.sheets['Sheet1']
worksheet.set_column('C:C', 15)
writer.save()
```

★ The code starts by opening the 'Data.txt' file in read mode and reading its contents into the data variable.

★ The data variable is split into a list using the separator " \n# " and then joined back into a single string with spaces in between.The string is split into a list again using spaces as the separator.

★ Two empty lists, time_list and carry_list, are initialized to store time values and carry values respectively.

- A loop processes the data list:
- If an item starts with 'Time=', the integer value following the '=' sign is extracted and added to time_list.

- If an item starts with 'Carry=', the value following the '=' sign is extracted and added to carry_list.

★ A pointer variable is set to 0 initially.Another loop iterates over time_list:

- If a time value is greater than 500, the pointer variable is set to the index of this value, and the loop breaks. This is used to find a specific threshold time.

★ Both time_list and carry_list are truncated, starting from the position determined by the pointer variable.

★ Carry delay calculation:

- x = int(bin_carry_list[i], 2) - int(bin_carry_list[i - 1], 2): It calculates the change in carry values between two consecutive time points by subtracting the previous carry value (converted from binary to decimal) from the current carry value.

- place = 0: place keeps track of the position of the bit being examined in the binary representation of x.

- while x > 0:: This loop goes through each bit of x until there are no more bits left to examine.

- if x & 1 == 1:: It checks if the least significant bit of x is 1. If true, it means a carry change occurred at this bit position.

- if place < len(Carry_Delay):: This ensures that the place value (bit position) is within the bounds of the Carry_Delay list to prevent errors.

- Carry_Delay[place] = time_list[i] - 500: If all conditions are met, it records the delay at the corresponding bit position in the Carry_Delay list. The delay is calculated by subtracting 500 from the current time value (time_list[i] - 500), indicating when the carry change happened at this bit position.

★ Exporting to Excel:

- The Carry_Delay list is used to create a Pandas DataFrame.
- An ExcelWriter object is created to write the DataFrame to an Excel file named 'Table_Assignment1.xlsx'.
- The DataFrame is written to the Excel sheet named 'Sheet1' starting from row 3 and column 2 (C).
- The 'C' column's width is set to 15.
- The Excel file is saved.

CODE USED TO CALCULATE SUM DELAY

C: > Users > VANAJA AGARWAL > Downloads > SumDataDelay.py > ...

```python
file = open('Sum.txt','r')

data =file.read()


data=data.split("\n")
data = " ".join(data)
data = data.split(" ")
time_list = []
carry_list = []

print(data)
for i in range(len(data)):
    if(i%2==0):
        time_list.append(int(data[i]))
    else:
        carry_list.append((data[i]))

print("Time List:", time_list)
print("Carry List:", carry_list)
#print(data)
pointer=0

print(len(time_list),len(carry_list))
hex_carry_list = [bin(int(carry, 2))[2:] for carry in carry_list]

Carry_Delay = [0] * 64
```

```
for i in range(1, len(hex_carry_list)):
    x = ~int(hex_carry_list[i], 2) - ~int(hex_carry_list[i - 1], 2)
    print(x)
    place = 0
    while x > 0:
        if x & 1 == 1:
            if place < len(Carry_Delay):  # Check if place is within the bounds
                Carry_Delay[place] = time_list[i] - 500
        x = x >> 1

        place += 1
    print(place)

print((Carry_Delay))
import pandas as pd


df = pd.DataFrame({'Carry_Delay': Carry_Delay})

writer = pd.ExcelWriter('Table_Assignment1.xlsx', engine='xlsxwriter')


df.to_excel(writer, sheet_name='Sheet1', startrow=3, startcol=5, index=False)

workbook = writer.book
worksheet = writer.sheets['Sheet1']


worksheet.set_column('C:C', 15)

writer.save()
```

ADVANTAGES OF 64 - BIT CLA

64-bit CLA Adder:

Advantages:

1. Speedy: Computes 64-bit sums quickly.
2. Parallel: Handles carry bits all at once, reducing delay.
3. Easy to design and efficient for chips.

Disadvantages:

1. Complex: Uses many gates, leading to higher power consumption.
2. Occupies more chip area.
3. Less practical for very wide additions due to complexity.

CLA adders are super-fast but can be complex and space-consuming for larger numbers.

Ripple Carry Adder:

Advantages:

1. Easy to design and use.
2. Works well for small additions.
3. Can be combined to build wider adders.

Disadvantages:

1. Becomes slower as numbers get larger.
2. Computes carry one after the other.

Ripple carry adders are simple and economical but not the fastest for large calculations.

REAL LIFE APPLICATIONS

1. CLA adders perform fast addition and subtraction operations, ideal for applications needing rapid calculations.

2. They calculate carry signals for all bits simultaneously, reducing delay and speeding up calculations.

3. Scalability and Bit-Wide Operations

5. Applications in High-Performance Computing: CLA adders are commonly used in supercomputers, GPUs, and specialized hardware for demanding tasks like scientific simulations and data processing.

6. Reduced Power Consumption.