ECEN 248 - Lab Report

Lab Number: 9

Lab Title: An Introduction to High-Speed Addition

Section Number: 513

Student's Name: Abhishek Bhattacharyya

Student's UIN: 731002289

Date: 4/13/2023

TA: Hesam Mazaheri

Objectives:

In the lab from last week, we discovered that the maximum clock speed for a synchronous circuit is directly impacted by the propagation delay through combinational logic. This week introduces carry-lookahead addition in the form of a block carry-lookahead adder. The carry look-ahead adder is a popular fast-adder circuit for two-operand addition. The required parts will be created using a combination of dataflow and structural Verilog, and will then be simulated in Vivado to test for functionality.

Design:

All of the deliverables for this lab were coded from the base code and instructions provided in the lab manual. A Generate/Propagate Unit, Carry-Lookahead Unit, and Summation Unit were all modeled in dataflow Verilog using the base modules given in the lab manual. Then, each module was combined to create a 4-bit Carry-lookahead adder and this was run on the test bench to check for errors.

For Experiment 2, template code was given to create a Block carry-lookahead unit. The code was completed to create a 2-level 16-bit carry-lookahead adder. Finally, the data from each simulation was used to find the propagation delay.

Verilog code for all modules coded and tested is provided below.

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This module describes the Carry Generate/Propagate
4    Unit for 4-bit carry-lookahead addition */
5 module generate_propagate_unit(G, P, X, Y);
6     // ports are wires as we will use dataflow
7     output wire [3:0] G, P;
8     input wire [3:0] X, Y;
9
10    // this assigns the generates following
11    // its boolean expression of Gi = XiYi
12    assign G[0] = X[0] & Y[0];
13    assign G[1] = X[1] & Y[1];
14    assign G[2] = X[2] & Y[2];
15    assign G[3] = X[3] & Y[3];
16
17    // this assigns the propagates following
18    // its boolean expression of Pi = Xi XOR Yi
19    assign P[0] = X[0] ^ Y[0];
20    assign P[1] = X[1] ^ Y[1];
21    assign P[2] = X[2] ^ Y[2];
22    assign P[3] = X[3] ^ Y[3];
23 endmodule
```

Figure 1: Source Code for the 4-bit GPU

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This module describes the 4-bit summation unit
4    for a carry-lookahead adder */
5 module summation_unit(S, P, C);
6     // ports are wires because we will use dataflow
7     output wire [3:0] S; // sum vector
8     input wire [3:0] P, C; // propagate and carry vectors
9
10    // here i assign each sum output to be the XOR
11    // of Pi and Ci which follows the boolean
12    // expression for the sum vectors
13    assign S[0] = P[0] ^ C[0];
14    assign S[1] = P[1] ^ C[1];
15    assign S[2] = P[2] ^ C[2];
16    assign S[3] = P[3] ^ C[3];
17 endmodule
```

Figure 2: Source Code for the 4-bit SU

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This module describes the 4-bit carry-lookahead unit
4    for a carry-lookahead adder */
5 module carry_lookahead_unit(C, G, P, C0);
6     // ports are wires because we will use dataflow
7     output wire [4:1] C; // C4, C3, C2, C1
8     input wire [3:0] G, P; // generates and propagates
9     input wire C0; // input carry
10
11    // this assigns each of the carry outputs to their
12    // boolean expression dealing with the generate
13    // and propagate inputs that are given
14    assign C[1] = G[0] | (P[0] & C0);
15    assign C[2] = G[1] | (P[1] & G[0]) | (P[1] & P[0] & C0);
16    // since this boolean expression is much longer than the
17    // other two, i split it up into 2 separate lines of
18    // code
19    assign C[3] = G[2] | (P[2] & G[1]) | (P[2] & P[1] & G[0])
20    | (P[2] & P[1] & P[0] & C0);
21    // same as the comment directly above this one
22    assign C[4] = G[3] | (P[3] & G[2]) | (P[3] & P[2] & G[1])
23    | (P[3] & P[2] & P[1] & G[0]) | (P[3] & P[2] & P[1] & P[0] & C0);
24 endmodule
```

Figure 3: Source Code for the 4-bit CLAU

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This is the top-level module for a 4-bit
4    carry-lookahead adder */
5 module carry_lookahead_4bit(Cout, S, X, Y, Cin);
6     // ports are wires as we will use structural
7     output wire Cout; // C_4 for a 4-bit adder
8     output wire [3:0] S; // final 4-bit sum vector
9     input wire [3:0] X, Y; // the 4-bit addends
10    input wire Cin; // input carry
11    wire [3:0] G, P;
12    wire [4:1] C;
13
14    // here i use the gpu, clu, and su modules that i created
15    // earlier and since the outputs are [3:0] in the modules,
16    // we have to follow the same format here in order for
17    // the simulation to work
18    generate_propagate_unit gpu(G[3:0], P[3:0], X[3:0], Y[3:0]);
19
20    // here i create the implementation of the clu for the
21    // final 4-bit carry-lookahead-adder
22    carry_lookahead_unit clu(C[4:1], G[3:0], P[3:0], Cin);
23
24    // here i assign Cout with the last output carry
25    // that's given above
26    assign Cout = C[4];
27
28    // here i need to include the initial input carry
29    // which is given as Cin, so i need to include that
30    // with the first 3 carries from the clu code
31    summation_unit su(S[3:0], P[3:0], {C[3:1],Cin});
32 endmodule
```

Figure 4: Source Code for the 4-bit CLA

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This module describes the Carry Generate/Propagate
4    Unit for 4-bit carry-lookahead addition */
5 module generate_propagate_unit(G, P, X, Y);
6     // ports are wires as we will use dataflow
7     output wire [3:0] G, P;
8     input wire [3:0] X, Y;
9
10    // this assigns the generates following
11    // its boolean expression of Gi = XiYi
12    assign #2 G[0] = X[0] & Y[0];
13    assign #2 G[1] = X[1] & Y[1];
14    assign #2 G[2] = X[2] & Y[2];
15    assign #2 G[3] = X[3] & Y[3];
16
17    // this assigns the propagates following
18    // its boolean expression of Pi = Xi XOR Yi
19    assign #2 P[0] = X[0] ^ Y[0];
20    assign #2 P[1] = X[1] ^ Y[1];
21    assign #2 P[2] = X[2] ^ Y[2];
22    assign #2 P[3] = X[3] ^ Y[3];
23 endmodule
```

Figure 5: Source Code for the 4-bit GPU with Delay

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This module describes the 4-bit summation unit
4    for a carry-lookahead adder */
5 module summation_unit(S, P, C);
6     // ports are wires because we will use dataflow
7     output wire [3:0] S; // sum vector
8     input wire [3:0] P, C; // propagate and carry vectors
9
10    // here i assign each sum output to be the XOR
11    // of Pi and Ci which follows the boolean
12    // expression for the sum vectors
13    assign #2 S[0] = P[0] ^ C[0];
14    assign #2 S[1] = P[1] ^ C[1];
15    assign #2 S[2] = P[2] ^ C[2];
16    assign #2 S[3] = P[3] ^ C[3];
17 endmodule
```

Figure 6: Source Code for the 4-bit SU with Delay

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This module describes the 4-bit carry-lookahead unit
4    for a carry-lookahead adder */
5 module carry_lookahead_unit(C, G, P, C0);
6     // ports are wires because we will use dataflow
7     output wire [4:1] C; // C4, C3, C2, C1
8     input wire [3:0] G, P; // generates and propagates
9     input wire C0; // input carry
10
11    // this assigns each of the carry outputs to their
12    // boolean expression dealing with the generate
13    // and propagate inputs that are given
14    assign #4 C[1] = G[0] | (P[0] & C0);
15    assign #4 C[2] = G[1] | (P[1] & G[0]) | (P[1] & P[0] & C0);
16    // since this boolean expression is much longer than the
17    // other two, i split it up into 2 separate lines of
18    // code
19    assign #4 C[3] = G[2] | (P[2] & G[1]) | (P[2] & P[1] & G[0])
20    | (P[2] & P[1] & P[0] & C0);
21    // same as the comment directly above this one
22    assign #4 C[4] = G[3] | (P[3] & G[2]) | (P[3] & P[2] & G[1])
23    | (P[3] & P[2] & P[1] & G[0]) | (P[3] & P[2] & P[1] & P[0] & C0);
24 endmodule
```

Figure 7: Source Code for the 4-bit CLAU with Delay

```
1 `timescale 1 ns/ 1 ps
2 `default_nettype none
3 /* This module describes the block carry-lookahead unit
4 for a 2-level carry-lookahead adder */
5 module block_carry_lookahead_unit(G_star, P_star, C, G, P, C0);
6     // ports are wires because we will use dataflow
7     output wire G_star, P_star; // block generate and propagate
8     output wire [3:1] C; // C3, C2, C1
9     input wire [3:0] G, P; // generates and propagates
10    input wire C0; // input carry
11
12    assign C[1] = G[0] | (P[0] & C0);
13    assign C[2] = G[1] | (P[1] & G[0]) | (P[1] & P[0] & C0);
14    // since this boolean expression is much longer than the
15    // other two, i split it up into 2 separate lines of
16    // code
17    assign C[3] = G[2] | (P[2] & G[1]) | (P[2] & P[1] & G[0])
18    | (P[2] & P[1] & P[0] & C0);
19
20    assign G_star = (G[3]) | (P[3] & G[2]) | (P[3] & P[2] & G[1])
21    | (P[3] & P[2] & P[1] & G[0]);
22    assign P_star = (P[3] & P[2] & P[1] & P[0]);
23 endmodule
```

Figure 8: Source Code for the 4-bit BLAU without Delay

```
`timescale 1 ns/ 1 ps
`default_nettype none
/* This module describes the block carry-lookahead unit
for a 2-level carry-lookahead adder */
module block_carry_lookahead_unit(G_star, P_star, C, G, P, C0);
    // ports are wires because we will use dataflow
    output wire G_star, P_star; // block generate and propagate
    output wire [3:1] C; // C3, C2, C1
    input wire [3:0] G, P; // generates and propagates
    input wire C0; // input carry

    assign #4 C[1] = G[0] | (P[0] & C0);
    assign #4 C[2] = G[1] | (P[1] & G[0]) | (P[1] & P[0] & C0);
    // since this boolean expression is much longer than the
    // other two, i split it up into 2 separate lines of
    // code
    assign #4 C[3] = G[2] | (P[2] & G[1]) | (P[2] & P[1] & G[0])
    | (P[2] & P[1] & P[0] & C0);

    assign #4 G_star = (G[3]) | (P[3] & G[2]) | (P[3] & P[2] & G[1])
    | (P[3] & P[2] & P[1] & G[0]);
    assign #2 P_star = (P[3] & P[2] & P[1] & P[0]);
endmodule
```

Figure 9: Source Code for the 4-bit BLAU with Delay

```
`timescale 1 ns/ 1 ps
`default_nettype none
/* This module describes the 16-bit summation unit
   for a carry-lookahead adder */
module summation_unit(S, P, C);
    // ports are wires because we will use dataflow
    output wire [15:0] S; // sum vector
    input wire [15:0] P, C; // propagate and carry vectors

    // here i assign each sum output to be the XOR
    // of Pi and Ci which follows the boolean
    // expression for the sum vectors
    assign S[0] = P[0] ^ C[0];
    assign S[1] = P[1] ^ C[1];
    assign S[2] = P[2] ^ C[2];
    assign S[3] = P[3] ^ C[3];
    assign S[4] = P[4] ^ C[4];
    assign S[5] = P[5] ^ C[5];
    assign S[6] = P[6] ^ C[6];
    assign S[7] = P[7] ^ C[7];
    assign S[8] = P[8] ^ C[8];
    assign S[9] = P[9] ^ C[9];
    assign S[10] = P[10] ^ C[10];
    assign S[11] = P[11] ^ C[11];
    assign S[12] = P[12] ^ C[12];
    assign S[13] = P[13] ^ C[13];
    assign S[14] = P[14] ^ C[14];
    assign S[15] = P[15] ^ C[15];
endmodule
```

Figure 10: Source Code for the 16-bit SU without Delay

```
`timescale 1 ns/ 1 ps
`default_nettype none
/* This module describes the 16-bit summation unit
   for a carry-lookahead adder */
module summation_unit(S, P, C);
    // ports are wires because we will use dataflow
    output wire [15:0] S; // sum vector
    input wire [15:0] P, C; // propagate and carry vectors

    // here i assign each sum output to be the XOR
    // of Pi and Ci which follows the boolean
    // expression for the sum vectors
    assign #2 S[0] = P[0] ^ C[0];
    assign #2 S[1] = P[1] ^ C[1];
    assign #2 S[2] = P[2] ^ C[2];
    assign #2 S[3] = P[3] ^ C[3];
    assign #2 S[4] = P[4] ^ C[4];
    assign #2 S[5] = P[5] ^ C[5];
    assign #2 S[6] = P[6] ^ C[6];
    assign #2 S[7] = P[7] ^ C[7];
    assign #2 S[8] = P[8] ^ C[8];
    assign #2 S[9] = P[9] ^ C[9];
    assign #2 S[10] = P[10] ^ C[10];
    assign #2 S[11] = P[11] ^ C[11];
    assign #2 S[12] = P[12] ^ C[12];
    assign #2 S[13] = P[13] ^ C[13];
    assign #2 S[14] = P[14] ^ C[14];
    assign #2 S[15] = P[15] ^ C[15];
endmodule
```

Figure 11: Source Code for the 16-bit SU with Delay

```
`timescale 1 ns/ 1 ps
`default_nettype none
/* This module describes the Carry Generate/Propagate
   Unit for 16-bit carry-lookahead addition */
module generate_propagate_unit(G, P, X, Y);
    // ports are wires as we will use dataflow
    output wire [15:0] G, P;
    input wire [15:0] X, Y;

    // this assigns the generates following
    // its boolean expression of Gi = XiYi
    assign #2 G[0] = X[0] & Y[0];
    assign #2 G[1] = X[1] & Y[1];
    assign #2 G[2] = X[2] & Y[2];
    assign #2 G[3] = X[3] & Y[3];
    assign #2 G[4] = X[4] & Y[4];
    assign #2 G[5] = X[5] & Y[5];
    assign #2 G[6] = X[6] & Y[6];
    assign #2 G[7] = X[7] & Y[7];
    assign #2 G[8] = X[8] & Y[8];
    assign #2 G[9] = X[9] & Y[9];
    assign #2 G[10] = X[10] & Y[10];
    assign #2 G[11] = X[11] & Y[11];
    assign #2 G[12] = X[12] & Y[12];
    assign #2 G[13] = X[13] & Y[13];
    assign #2 G[14] = X[14] & Y[14];
    assign #2 G[15] = X[15] & Y[15];

    // this assigns the propagates following
    // its boolean expression of Pi = Xi XOR Yi
    assign #2 P[0] = X[0] ^ Y[0];
    assign #2 P[1] = X[1] ^ Y[1];
    assign #2 P[2] = X[2] ^ Y[2];
    assign #2 P[3] = X[3] ^ Y[3];
    assign #2 P[4] = X[4] ^ Y[4];
    assign #2 P[5] = X[5] ^ Y[5];
    assign #2 P[6] = X[6] ^ Y[6];
    assign #2 P[7] = X[7] ^ Y[7];
    assign #2 P[8] = X[8] ^ Y[8];
    assign #2 P[9] = X[9] ^ Y[9];
    assign #2 P[10] = X[10] ^ Y[10];
    assign #2 P[11] = X[11] ^ Y[11];
    assign #2 P[12] = X[12] ^ Y[12];
    assign #2 P[13] = X[13] ^ Y[13];
    assign #2 P[14] = X[14] ^ Y[14];
    assign #2 P[15] = X[15] ^ Y[15];
endmodule
```

Figure 12: Source Code for the 16-bit GPU with Delay

```
`timescale 1 ns/ 1 ps
`default_nettype none
/* This is the top-level module for a 16-bit 2-level
carry-lookahead adder */
module carry_lookahead_16bit(Cout, S, X, Y, Cin);
    // ports are wires as we will use structural
    output wire Cout; // C_16 for a 16-bit adder
    output wire [15:0] S; // final 16-bit sum vector
    input wire [15:0] X, Y; // the 16-bit addends
    input wire Cin; // input carry
    // intermediate nets
    wire [16:0] C; // 5-bit carry vector
    wire [15:0] P, G; // generate and propagate vectors
    wire [3:0] P_star, G_star; // block generates
                               // and propagates
    // hook up input and output carry
    assign C[0] = Cin;
    assign Cout = C[16];
    // sub-modules
    // instantiate the generate/propagate unit here
    generate_propagate_unit gpu(G[15:0], P[15:0], X[15:0], Y[15:0]);
    /* for the more complicated module instantiations
    you will probably find this form easier to read */
    block_carry_lookahead_unit BCLAU0(
        /* since we are being explicit with our ports,
        the order does not matter */
        /* for BCLAU0, we need to use the first 4 wires for
        the generate/propagate and the last 3 for the carry
        which makes the first carry our carry in */
        .G_star (G_star[0]),
        .P_star (P_star[0]),
        .C (C[3:1]),
        .G (G[3:0]),
        .P (P[3:0]),
        .C0 (C[0])
    ); // a little more verbose, but much easier to follow
    // instantiate BCLAUs 1-3 here
    block_carry_lookahead_unit BCLAU1(
        .G_star (G_star[1]),
        .P_star (P_star[1]),
        .C (C[7:5]),
        .G (G[7:4]),
        .P (P[7:4]),
        .C0 (C[4])
    );
    block_carry_lookahead_unit BCLAU2(
        .G_star (G_star[2]),
        .P_star (P_star[2]),
        .C (C[11:9]),
        .G (G[11:8]),
        .P (P[11:8]),
        .C0 (C[8])
    );
    block_carry_lookahead_unit BCLAU3(
        .G_star (G_star[3]),
        .P_star (P_star[3]),
        .C (C[15:13]),
        .G (G[15:12]),
        .P (P[15:12]),
        .C0 (C[12])
    );
    // we will use the same form for this one too
    carry_lookahead_unit CLAU(
        .C ({C[16], C[12], C[8], C[4]}),
        .G (G_star),
        .P (P_star),
        .C0 (C[0])
    );
    // instantiate the summation unit here
    summation_unit su(S[15:0], P[15:0], {C[15:1],Cin});
endmodule
```

Figure 13: Source Code for the 16-bit CLA

```
`timescale 1ns / 1ps
`define NUM_TESTS 1024
`define STRLEN 64
`define TEST_DELAY 40

module cla_16bit_tb;
  /*A task is similar to a procedure in the traditional programming language*/
  /*This particular task simply checks the output of our circuit against a
    known answer and prints a message based on the outcome. Additionally,
    this task increments the variable we are using to keep track of the
    number of tests successfully passed.*/
  task passTest;
      input [16:0] actualOut, expectedOut;
      input [`STRLEN*8:0] testType;
      inout [7:0] passed;

      if(actualOut == expectedOut) begin $display ("%s passed", testType); passed = passe
      else $display ("%s failed: %s should be %s", testType, actualOut, expectedOut);
  endtask
  /*this task simply informs the user of the final outcome of the test*/
  task allPassed;
      input [7:0] passed;
      input [7:0] numTests;

      if(passed == numTests) $display ("All tests passed");
      else $display("Some tests failed");
  endtask
  // Inputs
  reg [15:0] X;
  reg [15:0] Y;
  reg Cin;
  // Outputs
  wire Cout;
  wire [15:0] S;
  //test bench nets
  reg [16:0] Result;
  reg [7:0] passed;

  carry_lookahead_16bit uut (
      .Cout(Cout),
      .S(S),
      .X(X),
      .Y(Y),
      .Cin(Cin)
  );
  initial begin
      // Initialize Inputs
      X = 0;
      Y = 0;
      Cin = 0;
      passed = 0;
      repeat(`NUM_TESTS)
      begin
          #(`TEST_DELAY);
          Result = X + Y + Cin;
          passTest({Cout, S}, {Result}, "16-bit Carry-lookahead Adder Unit Test", passed);
          X = $random;
          Y = $random;
          Cin = $random;
      end
      /*check to see if all tests passed*/
    allPassed(passed,`NUM_TESTS);
    $stop; //halt simulation cause we are done!
  end
endmodule
```

Figure 14: 16-bit CLA Test Bench

Results:

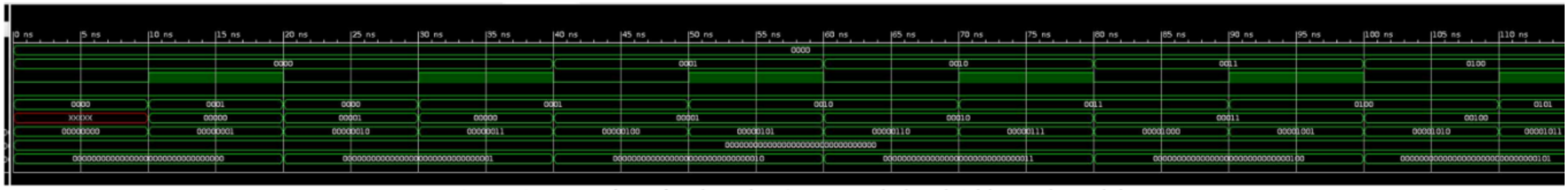

Figure 15: *Waveform for the 4-bit Carry-Lookahead Adder with no delay*

```
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
INFO: [USF-XSim-96] XSim completed. Design snapshot '
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Figure 16: *Console Output for the 4-bit CLA with no delay*

```
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 0f should be 0d
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 09 should be 0f
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 13 should be 11
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 0a should be 04
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 04 should be 0c
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 0d should be 05
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
INFO: [USF-XSim-96] XSim completed. Design snapshot 'cla_4bit
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Figure 17: *Console Output for the 4-bit CLA with 6 ns delay*

Figure 18: *Waveform for the 4-bit CLA with 6 ns delay*

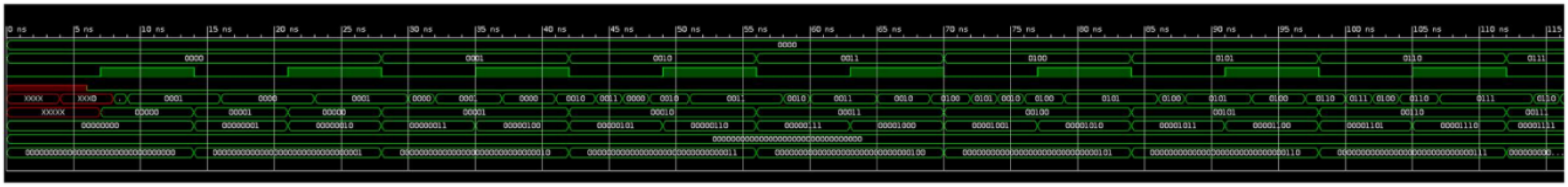

Figure 19: *Waveform for the 4-bit CLA with 7 ns delay*

```
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 0f should be 03
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 07 should be 05
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 01 should be 07
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 0b should be 09
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 05 should be 0b
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 0f should be 0d
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 09 should be 0f
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 13 should be 11
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test failed: 0a should be 04
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
INFO: [USF-XSim-96] XSim completed. Design snapshot 'cla_4b
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Figure 20: *Console Output for the 4-bit CLA with 7 ns delay*

```
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
INFO: [USF-XSim-96] XSim completed. Design snapshot '
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Figure 21: *Console Output for the 4-bit CLA with 8 ns delay*

Figure 22: *Waveform for the 4-bit CLA with 8 ns delay*

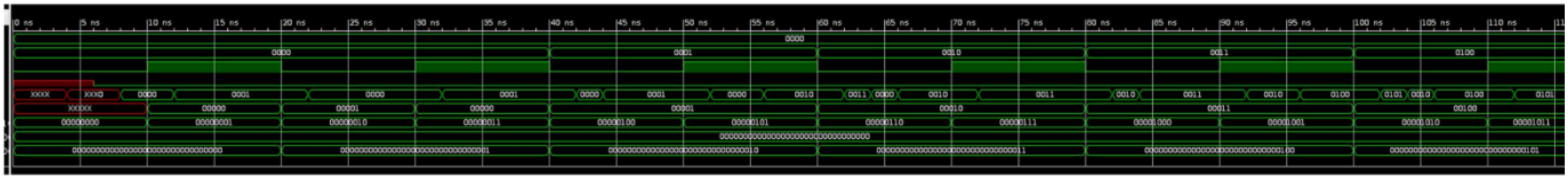

Figure 23: *Waveform for the 4-bit CLA with initial delay (10 ns)*

```
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
Carry-lookahead Adder Unit Test passed
INFO: [USF-XSim-96] XSim completed. Design snapshot '
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Figure 24: *Console Output for the 4-bit CLA with initial delay*

```
# }
# run 1000ns
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
            16-bit Carry-lookahead Adder Unit Test passed
INFO: [USF-XSim-96] XSim completed. Design snapshot 'cla_16bit_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Figure 25: *Console Output for the 16-bit CLA with no delay*

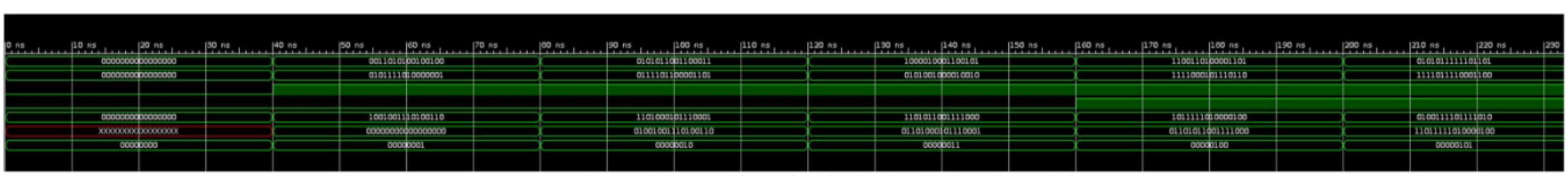

Figure 26: *Waveform for the 16-bit CLA with no delay*

Figure 27: *Waveform for the 16-bit CLA with 15 ns delay*

```
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 0dbf2 should be 0bbf2
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 134cb should be 114cb
    16-bit Carry-lookahead Adder Unit Test failed: 104e2 should be 10602
    16-bit Carry-lookahead Adder Unit Test failed: 15f9c should be 15f7c
    16-bit Carry-lookahead Adder Unit Test failed: 0d1c7 should be 0dfe7
    16-bit Carry-lookahead Adder Unit Test failed: 0ff17 should be 0fff7
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 0c707 should be 0e707
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 14b68 should be 14b88
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 12fce should be 14dce
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 165f3 should be 185f3
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 17cd3 should be 15cd3
    16-bit Carry-lookahead Adder Unit Test failed: 043a4 should be 063a4
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 1993b should be 1991b
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 16877 should be 16a77
    16-bit Carry-lookahead Adder Unit Test failed: 0f039 should be 0f019
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 0fe46 should be 0de46
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 0b74b should be 0b54b
    16-bit Carry-lookahead Adder Unit Test failed: 0c01d should be 0c21d
    16-bit Carry-lookahead Adder Unit Test failed: 0ff56 should be 0dd56
    16-bit Carry-lookahead Adder Unit Test failed: 16036 should be 18036
    16-bit Carry-lookahead Adder Unit Test failed: 0ac26 should be 0ae26
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 16df7 should be 16bd7
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 0fdf0 should be 0fbf0
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test failed: 06742 should be 06762
    16-bit Carry-lookahead Adder Unit Test failed: 13c5a should be 11c5a
Sim completed. Design snapshot 'cla_16bit_tb_behav' loaded.
Sim simulation ran for 1000ns
```

```
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
    16-bit Carry-lookahead Adder Unit Test passed
(Sim completed. Design snapshot 'cla_16bit_tb_behav' loade
(Sim simulation ran for 1000ns
```

Figure 28: *Console Output for the 16-bit CLA with 15 ns delay*

Figure 29: *Console Output for the 16-bit CLA with 16 ns delay*

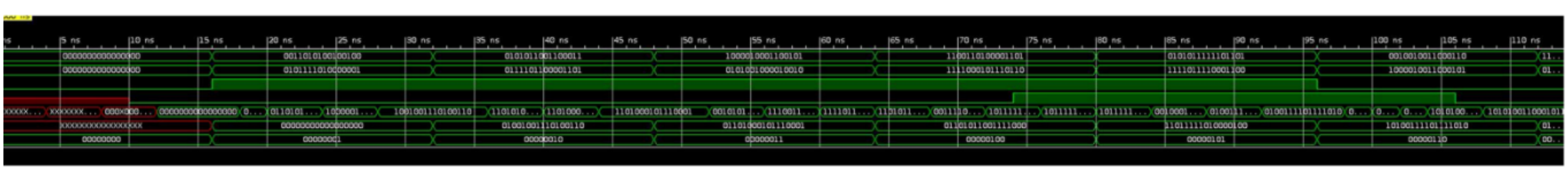

Figure 30: *Waveform for the 16-bit CLA with 16 ns delay*

Conclusion:

To complete this lab, I was able to use my knowledge of how to run and set up Vivado to build and simulate an advanced, high-speed carry-lookahead adder. First, the code was written with all modules required to build the 4-bit and 16-bit CLAs. Then, the code was tested for functionality and waveforms generated. Finally, each design was tested with a number of different delays to optimize the runtime and find the true number of gate delays in the circuit. This allowed me to illustrate how gate delays affect a circuit and find what the maximum theoretical clock speed of the circuit is before it starts causing errors.

Post-Lab Deliverables:

1. ***Include the source code with comments for all modules in lab. You do not have to include test bench code. Code without comments will not be accepted!***

All code for this assignment is included in the **Design** section.

2. ***Include the simulation screenshots requested in the above experiments in addition to the corresponding test bench console output.***

All simulation output with waveforms and console output is included in the **Results** section.

Experiment Part 1 2.b: Re-simulate the test bench and measure the propagation delay. To do this: i. Open the test bench file. Scroll down until you find the for loop. ii. Find the two lines in this loop that just read "#10;". iii. Change the delay to a low value such as 6 in both lines. iv. Re-simulate the test bench. If every test still passes, ensure your delays are added correctly. v. Increment these two delay values until all tests pass again. The final delay is your measured delay.

Following these steps, the value I got for the final delay (measured delay) was **7ns**.

Experiment Part 2, 3.a: Modify the 16-bit carry-lookahead adder sub-modules to include 2 ns gate-delays. Set the TEST_DELAY define within the test bench to the delay you computed for the 2-level carry-lookahead adder. Re-simulate the test bench. Now, measure the propagation delay of your adder by decrementing TEST_DELAY and re-simulating until some tests fail.

In Figure 14, it is shown the initial value of TEST_DELAY in the test bench is 40 ns. I changed this value to what I calculated as the ideal delay in the prelab (16ns) and the tests ran properly with all tests passing. I then tried changing TIME_DELAY to 15ns and some tests failed, showing the best time for this adder design to run is 16ns.

So, **16ns** is the overall propagation delay for this 16-bit CLA design.

Experiment Part 2, 3.b: If your calculations in the prelab are correct and you correctly added delays to your sub-modules, you should find that the computed delay matches the measure delay. Is this the case?

In the prelab, I found the gate delay for just the adder to run is 16ns, which perfectly matches what was found in the lab. I did note that the complete delay is 17ns, but this takes into account the final c_{16} carry value which is not the sum and just the overflow bit for carrying to another adder in the chain. So, my calculations were correct.

4. How does the gate-count of the 16-bit carry-lookahead adder compare to that of a ripple-carry adder of the same size? Give gate counts for both.

The 16-bit CLA has total 82 gates while the 16-bit ripple-carry adder has total 80 gates. Despite having more gates, the CLA is much more efficient than the RCA, so it runs faster as it parallelizes the workload so multiple gates can run at the time same time (this parallelization is implemented using the GPU and Summation units).

5. How does the propagation delay of the 4-bit carry-lookahead adder compare to that of a ripple-carry adder of the same size? Give delay values for both.

The 4-bit CLA has an overall delay of 4 times the base delay (8ns) while the overall delay of the 4-bit ripple-carry adder is 8 times the base gate delay (16ns). This is despite the 4-bit CLA having 26 gates while 4-bit ripple carry adder has 20 gates. The 4-bit carry-lookahead adder works significantly faster since the CLA has multiple different units carrying operations out in parallel, greatly speeding up addition.

6. Similarly, how does the propagation delay of the 16-bit carry-lookahead adder compare to that of a ripple-carry adder of the same size? Give delay values for both.

The 16-bit CLA has a propagation delay of 8 times the base gate delay, while the 16-bit ripple carry adder has a propagation delay of about 32 times the base gate delay. The 16-bit CLA's overall delay is 16 ns while the overall delay for the 16-bit ripple-carry adder is 64 ns. This is despite the 16-bit carry lookahead adder having 82 gates in it while a 16-bit ripple carry adder consists of around 80 gates. The 16 bit carry-lookahead adder is significantly faster because it is broken up into multiple units and many operations run in parallel, greatly speeding up addition.

7. Compare the delay growth of a ripple-carry adder versus a carry-lookahead adder. (ie. how does the delay increase as we increase the size of each adder?)

For the 16-bit ripple-carry adder, it appears the delay quadruples as the size increases. For example, with a 4-bit ripple carry adder, the propagation delay is 16 ns, which is 8*2, however, with the 16-bit ripple-carry adder, we can see that there's a propagation delay of 64 ns, which is 8*2*2. Therefore, we can expect a propagation delay of 128 ns for a 32-bit ripple-carry adder with a gate delay of 2ns.

For a CLA, the delay increases logarithmically - every time the adder doubles in bit width, the delay is roughly $4*2^{n/2}$. For example, the 4-bit CLA has a delay of 8ns (4*2). With the 16-bit CLA, the propagation delay is 16 ns (4*2*2). Therefore, for a 64-bit CLA, we can expect a delay of (4*2*2*2) = 32ns.

Overall, for a ripple-carry adder, the delay increases proportionally as the bit width of the adder increases. For a carry-lookahead adder, the delay increases at a significantly slower rate than with the ripple-carry adder- this makes sense since the CLA is a much more efficient design.

Important Student Feedback

1. What did you like most about the lab assignment and why? What did you like least about it and why?

What I liked most about this lab assignment was that I was able to keep practicing with Verilog, as I appreciate gaining more much-needed experience. It was also very fulfilling seeing my calculated gate delay match with the simultion. What I most disliked was just the general tedium of having to troubleshoot my code.

2. Were there any sections of the lab manual that were unclear? If so, what was unclear? Do you have any suggestions for improving the clarity?

The lab manual was clear and concise to understand.

3. What suggestions do you have to improve the overall lab assignment?

I do not have any suggestions to improve the lab assignment.