Assignment 1

Mohammad Hatif Osmani
002210701084

- **Full Adder using Half Adder (Structural & Behavioral)**
- **4-bit Ripple-Carry Adder (Structural & Behavioral)**
- **4-bit Carry-Lookahead Adder**
- **4-bit Carry-Skip Adder**
- **4-bit Carry-Save Adder**
- **4-bit BCD Adder**

1. Full Adder using Half Adder

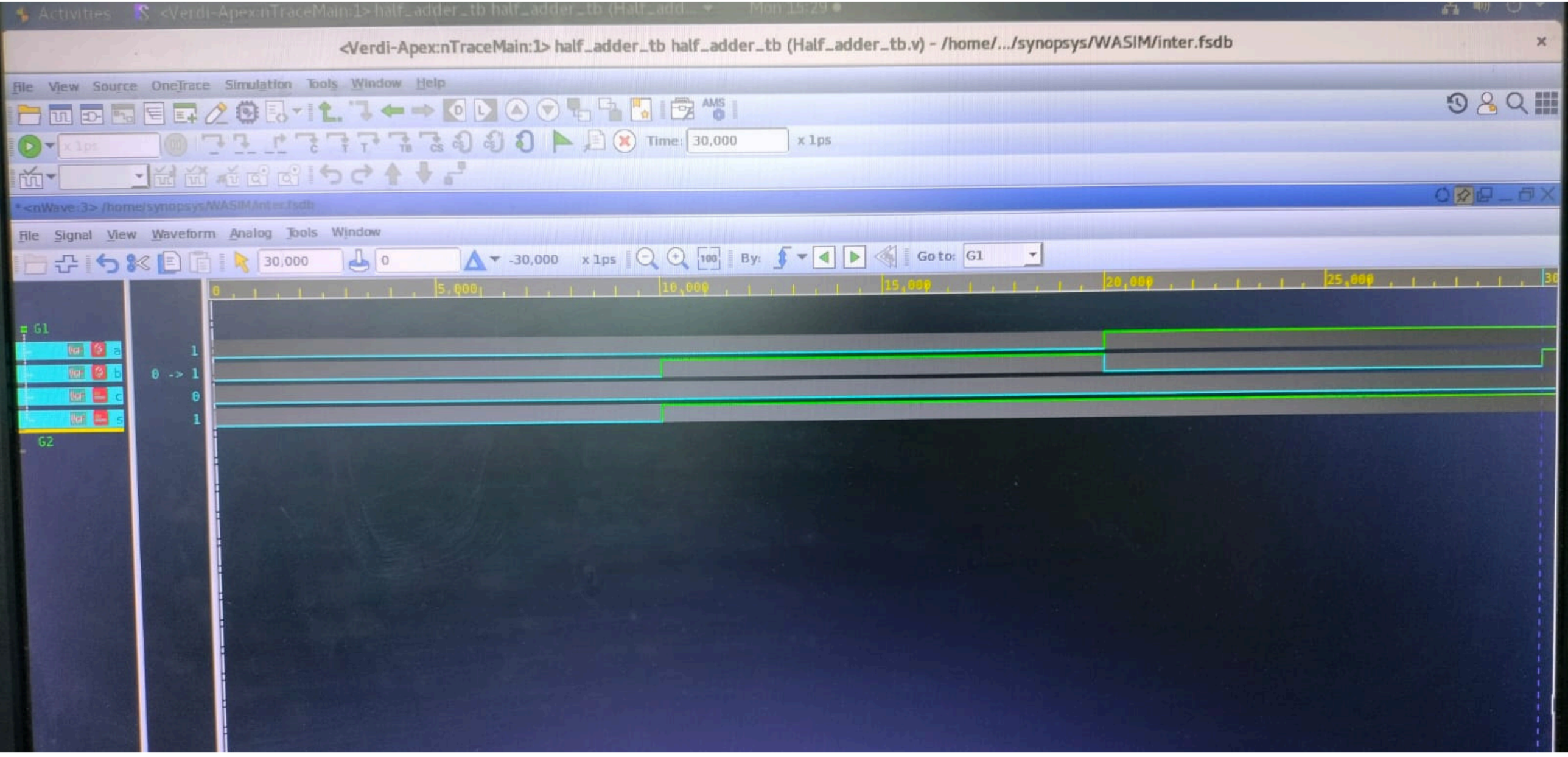
Concept: A Full Adder (FA) adds three 1-bit inputs (A, B, Cin) to produce a 2-bit output (Sum, Cout). It can be constructed from two Half Adders (HA) and an OR gate.

Logic:

- **Sum = A ⊕ B ⊕ Cin**
- **Cout = (A · B) + (Cin · (A ⊕ B))**

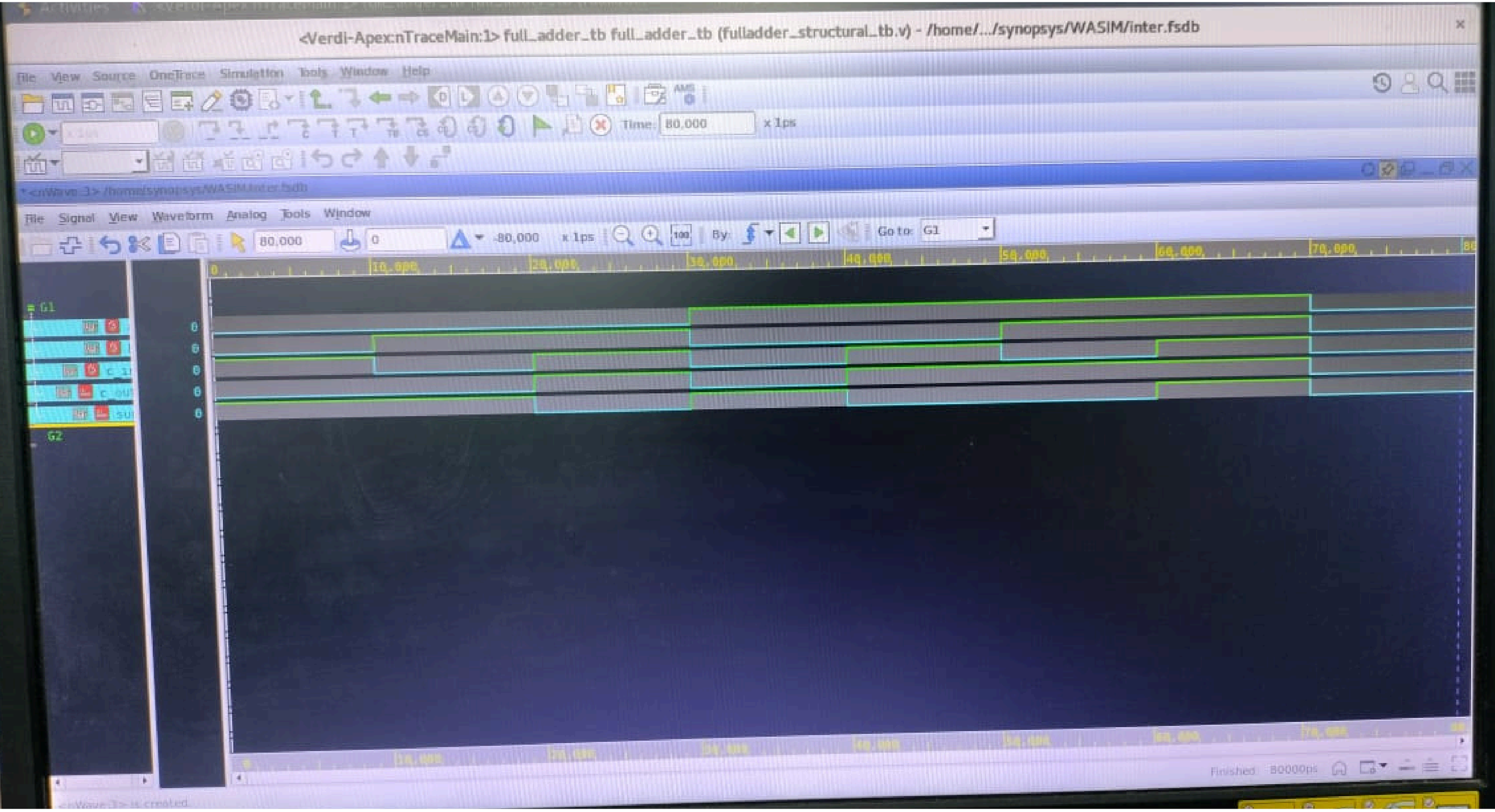
1.1 Verilog Code: Half Adder (Primitive) :This is the basic building block.

```
// half_adder.v
`timescale 1ns/1ps

module half_adder (
    input  wire a, b,
    output wire sum, carry
);
    // Dataflow assignments for sum and carry
    assign sum = a ^ b;   // XOR for sum
    assign carry = a & b; // AND for carry

endmodule
```


Activities
<Verdi-Apex:nTraceMain:1> half_adder_tb half_adder_tb (Half_adder_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 30,000
x 1ps
*<nWave:3> /home/synopsys/WASIM/inter.fsdb
File Signal View Waveform Analog Tools Window
30,000
0
-30,000
x 1ps
By:
Go to: G1
0
5,000
10,000
15,000
20,000
25,000
G1
a
b
c
s
G2
1
0 -> 1
0
1

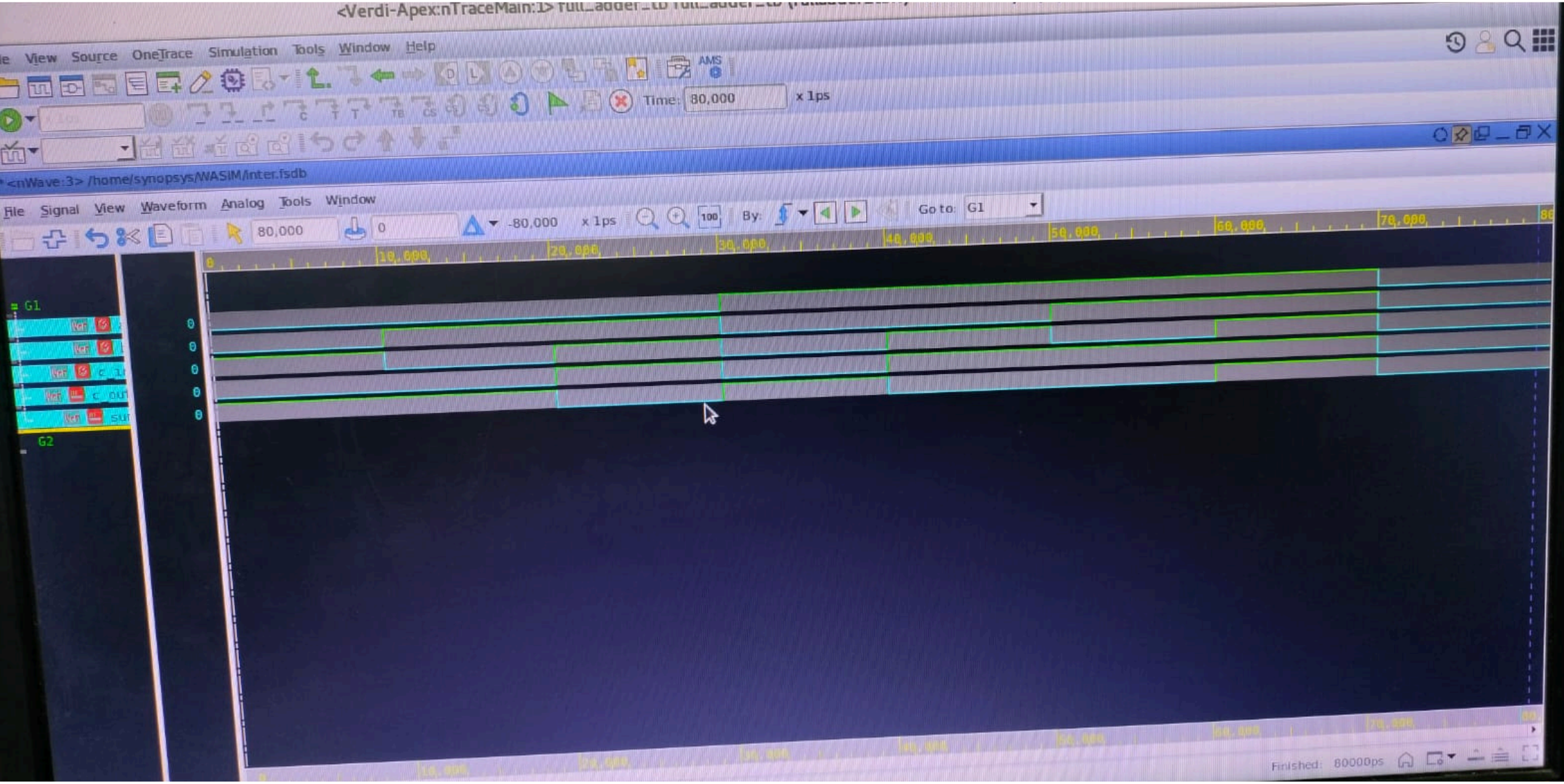
1.2 Verilog Code: Full Adder (Structural) : This model instantiates two half_adder modules.

```
// full_adder_structural.v
`timescale 1ns/1ps
`include "half_adder.v" // Include the half adder file

// Module for a Full Adder using a structural model
module full_adder_structural (
    input  wire a, b, cin,
    output wire sum, cout
);

    // Internal wires to connect the half adders
    wire s1, c1, c2;

    // Instantiate the first half adder
    // It adds the primary inputs a and b
    half_adder HA1 ( .a(a),  .b(b),  .sum(s1), .carry(c1));

    // Instantiate the second half adder
    // It adds the sum from the first HA (s1) and the carry-in (cin)
    half_adder HA2 ( .a(s1), .b(cin), .sum(sum), .carry(c2));

    // The final carry-out is the OR of the carries from both half adders
    assign cout = c1 | c2;

endmodule
```


<Verdi-Apex:nTraceMain:1> full_adder_tb full_adder_tb (fulladder_structural_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 80,000
x 1ps
File Signal View Waveform Analog Tools Window
80,000
-80,000
x 1ps
By:
Go to: G1
G1
G2
Finished 80000ps

1.3 Verilog Code: Full Adder (Behavioral): This model describes the function using arithmetic operators.

```
// full_adder_behavioral.v
// Module for a Full Adder using a behavioral model
module full_adder_behavioral (
    input  wire a, b, cin,
    output wire sum, cout
);
    // The concatenation {cout, sum} is a 2-bit value.
    // The sum of three 1-bit numbers (a, b, cin) can be 0, 1, 2, or 3,
    // which fits perfectly into a 2-bit result.
    assign {cout, sum} = a + b + cin;

endmodule
```


<Verdi-Apex:nTraceMain:1> full_adder_tb full_adder_tb
View Source OneTrace Simulation Tools Window Help
Time: 80,000 x 1ps
<nWave:3> /home/synopsys/WASIM/inter.fsdb
File Signal View Waveform Analog Tools Window
80,000
0
-80,000 x 1ps
By:
Go to: G1
10,000
20,000
30,000
40,000
50,000
60,000
70,000
G1
G2
Finished: 80000ps

1.4 Verilog Code: Full Adder Testbench: This testbench verifies both structural and behavioral models against all 8 possible input combinations.

```
// tb_full_adder.v
`timescale 1ns/1ps

module tb_full_adder;
   reg a, b, cin;
   wire sum_s, cout_s; // For structural model
   wire sum_b, cout_b; // For behavioral model

   full_adder_structural UUT_s (.*);
   full_adder_behavioral UUT_b (.*);

   initial begin
      $display("Time\t a b cin | cout_s sum_s | cout_b sum_b");
      $monitor("%2dns\t %b %b %b |   %b     %b   |   %b     %b",
            $time, a, b, cin, cout_s, sum_s, cout_b, sum_b);

      {a, b, cin} = 3'b000; #10; {a, b, cin} = 3'b001; #10;
      {a, b, cin} = 3'b010; #10; {a, b, cin} = 3'b011; #10;
      {a, b, cin} = 3'b100; #10; {a, b, cin} = 3'b101; #10;
      {a, b, cin} = 3'b110; #10; {a, b, cin} = 3'b111; #10;
      $finish;
   end
endmodule
```

2. 4-bit Ripple-Carry Adder (RCA)

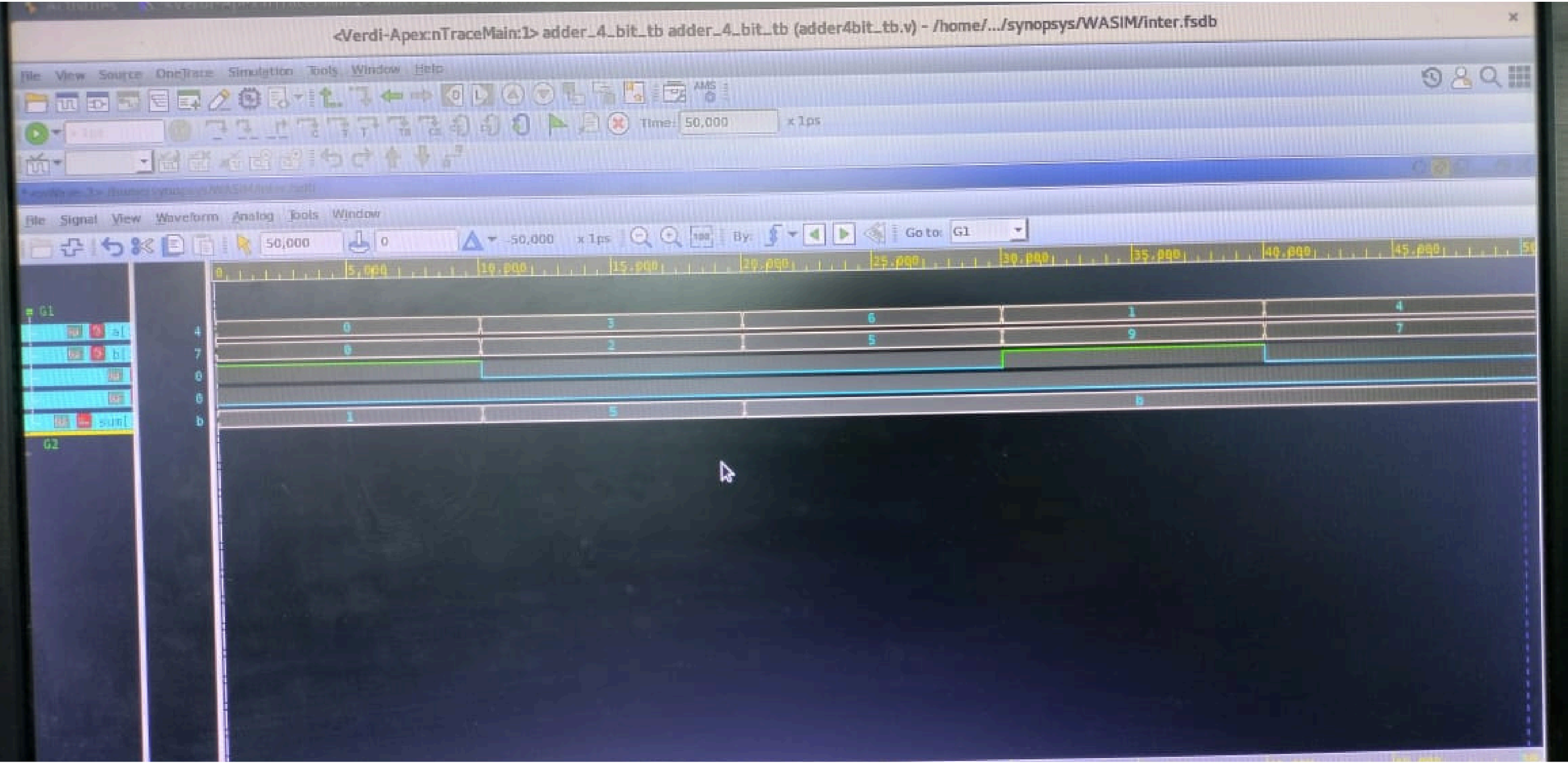
Concept: A 4-bit adder built by chaining four Full Adders. The carry-out of one stage becomes the carry-in of the next, causing the carry to "ripple" from the LSB to the MSB.
Characteristic: Simple to design, but slow for large bit-widths due to carry propagation delay.

Logic (for each bit i):

- **Sum[i] = A[i] ⊕ B[i] ⊕ C[i]**
- **C[i+1] = (A[i] · B[i]) + (C[i] · (A[i] ⊕ B[i]))**
- **Where C[0] is the primary Cin.**

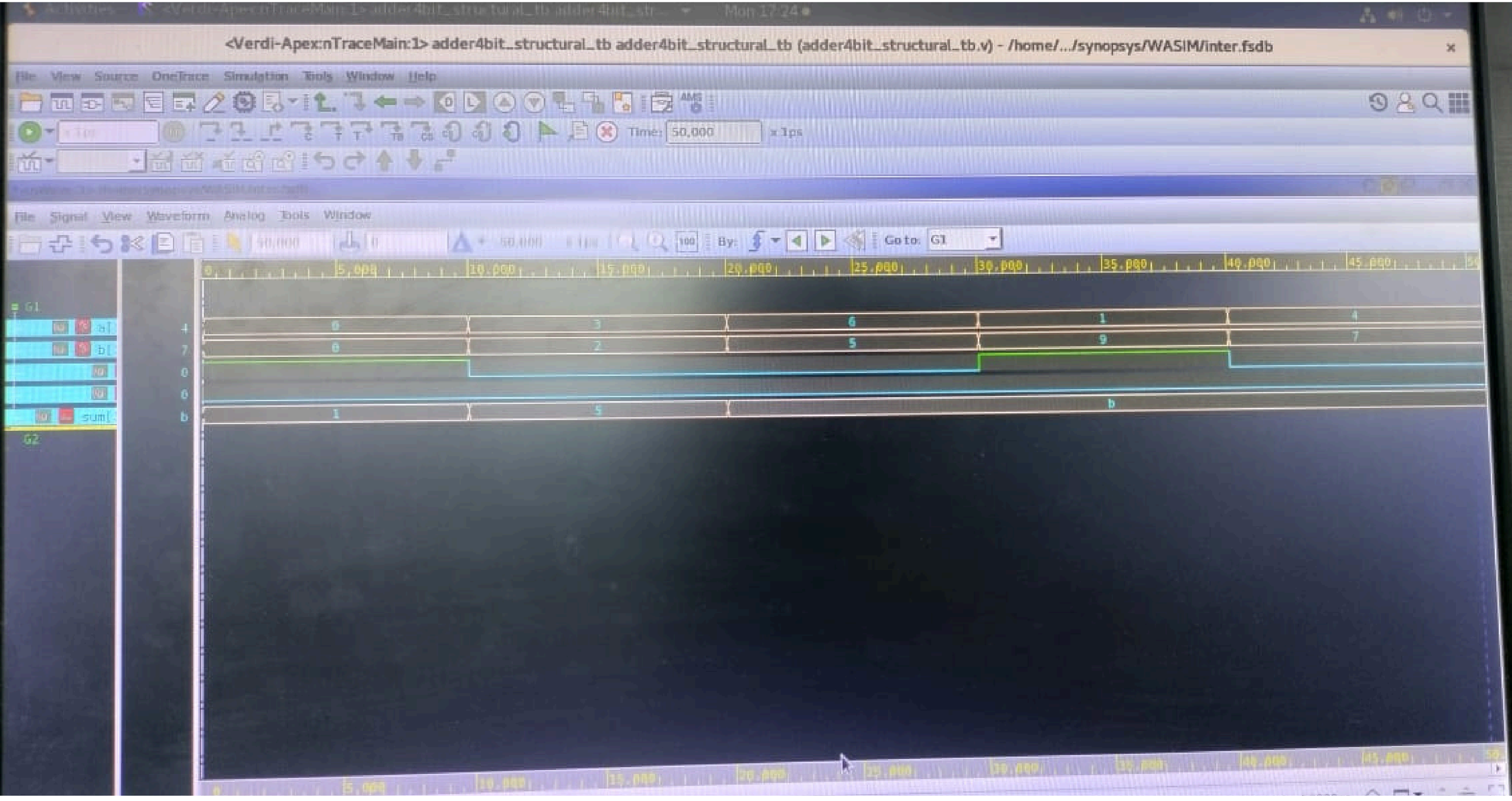
2.1 Verilog Code: 4-bit RCA (Behavioral): The + operator infers an adder. This is the most common and efficient way to model it.

```
// four_bit_adder_behavioral.v
`timescale 1ns/1ps

module four_bit_adder_behavioral (
    input  wire [3:0] A, B,
    input  wire       Cin,
    output wire [3:0] Sum,
    output wire       Cout
);
    assign {Cout, Sum} = A + B + Cin;

endmodule
```


<Verdi-Apex:nTraceMain:1> adder_4_bit_tb adder_4_bit_tb (adder4bit_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 50,000
x 1ps
File Signal View Waveform Analog Tools Window
50,000
-50,000
Go to: G1
G1
G2
sum
5,000
10,000
15,000
20,000
25,000
30,000
35,000
40,000
45,000

2.2 Verilog Code: 4-bit RCA (Structural): This model instantiates four of the full_adder_structural modules from Part 1.

```
// four_bit_adder_structural.v
`timescale 1ns/1ps
`include "full_adder_structural.v"
module four_bit_adder_structural (
    input  wire [3:0] A, B, input wire Cin,
    output wire [3:0] Sum, output wire Cout
);
    wire C1, C2, C3;
    // Instantiate four full adders, chaining the carry bit
    full_adder_structural FA0 (A[0], B[0], Cin, Sum[0], C1);
    full_adder_structural FA1 (A[1], B[1], C1,  Sum[1], C2);
    full_adder_structural FA2 (A[2], B[2], C2,  Sum[2], C3);
    full_adder_structural FA3 (A[3], B[3], C3,  Sum[3], Cout);
endmodule
```


<Verdi-Apex:nTraceMain:1> adder4bit_structural_tb adder4bit_structural_tb (adder4bit_structural_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 50,000
x 1ps
File Signal View Waveform Analog Tools Window
By:
Go to: G1
G1
a[
b[
sum[
G2
5,000
10,000
15,000
20,000
25,000
30,000
35,000
40,000
45,000
0
3
6
1
4
0
2
5
9
7
1
5
b

2.3 Verilog Code: 4-bit RCA Testbench:

```
// tb_four_bit_adder.v
`timescale 1ns/1ps

module tb_four_bit_adder;
   reg  [3:0] A, B; reg Cin;
   wire [3:0] Sum; wire Cout;
   four_bit_adder_structural UUT (.*);
   initial begin
      $display("Time\t A    +    B   + Cin = Cout Sum");
      $monitor("%2dns\t %h(h) + %h(h) + %b   =   %b   %h(h)",
            $time, A, B, Cin, Cout, Sum);

      A = 4'h3; B = 4'h4; Cin = 1'b0; #10; // 7
      A = 4'h8; B = 4'h9; Cin = 1'b0; #10; // 17 -> 1 0001
      A = 4'hF; B = 4'h1; Cin = 1'b0; #10; // 16 -> 1 0000
      A = 4'hF; B = 4'hF; Cin = 1'b1; #10; // 31 -> 1 1111
      $finish;
   end
endmodule
```

3. 4-bit Carry-Lookahead Adder (CLA)

Concept: A high-speed adder that computes carries in parallel, eliminating the ripple delay. It uses Propagate (P) and Generate (G) signals.

Logic:

- **G[i] = A[i] · B[i] (A carry is generated at bit i)**
- **P[i] = A[i] ⊕ B[i] (A carry-in propagates through bit i)**
- **C[i+1] = G[i] + (P[i] · C[i])**
- **Sum[i] = P[i] ⊕ C[i]**

3.1 Verilog Code: 4-bit CLA Design

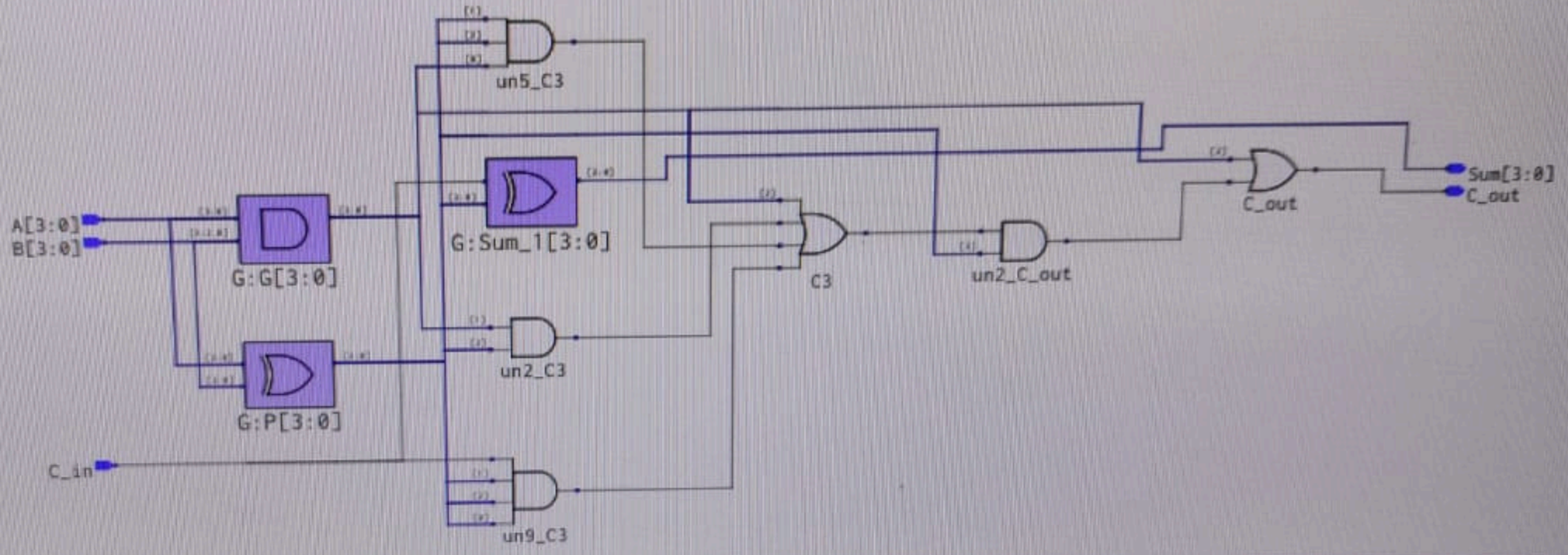
```
// carry_lookahead_adder.v
`timescale 1ns/1ps

module carry_lookahead_adder (
    input  wire [3:0] A, B, input  wire Cin,
    output wire [3:0] Sum, output wire Cout
);
    wire [3:0] P, G;
    wire [4:0] C;

    // Propagate and Generate signals
    assign P = A ^ B;
    assign G = A & B;

    // Parallel carry calculation
    assign C[0] = Cin;
    assign C[1] = G[0] | (P[0] & C[0]);
    assign C[2] = G[1] | (P[1] & (G[0] | (P[0] & C[0])));
    assign C[3] = G[2] | (P[2] & C[2]);
    assign C[4] = G[3] | (P[3] & C[3]);

    assign Sum = P ^ C[3:0];
    assign Cout = C[4];
endmodule
```

Synplify (R) Base V-2023.09-SP1 - [/home/synopsys/rev_80/synwork/bcd_encoder_mult.srs]

Project Run Analysis HDL-Analyst Options Window Tech-Support Web Help

Schematic Option

(3)



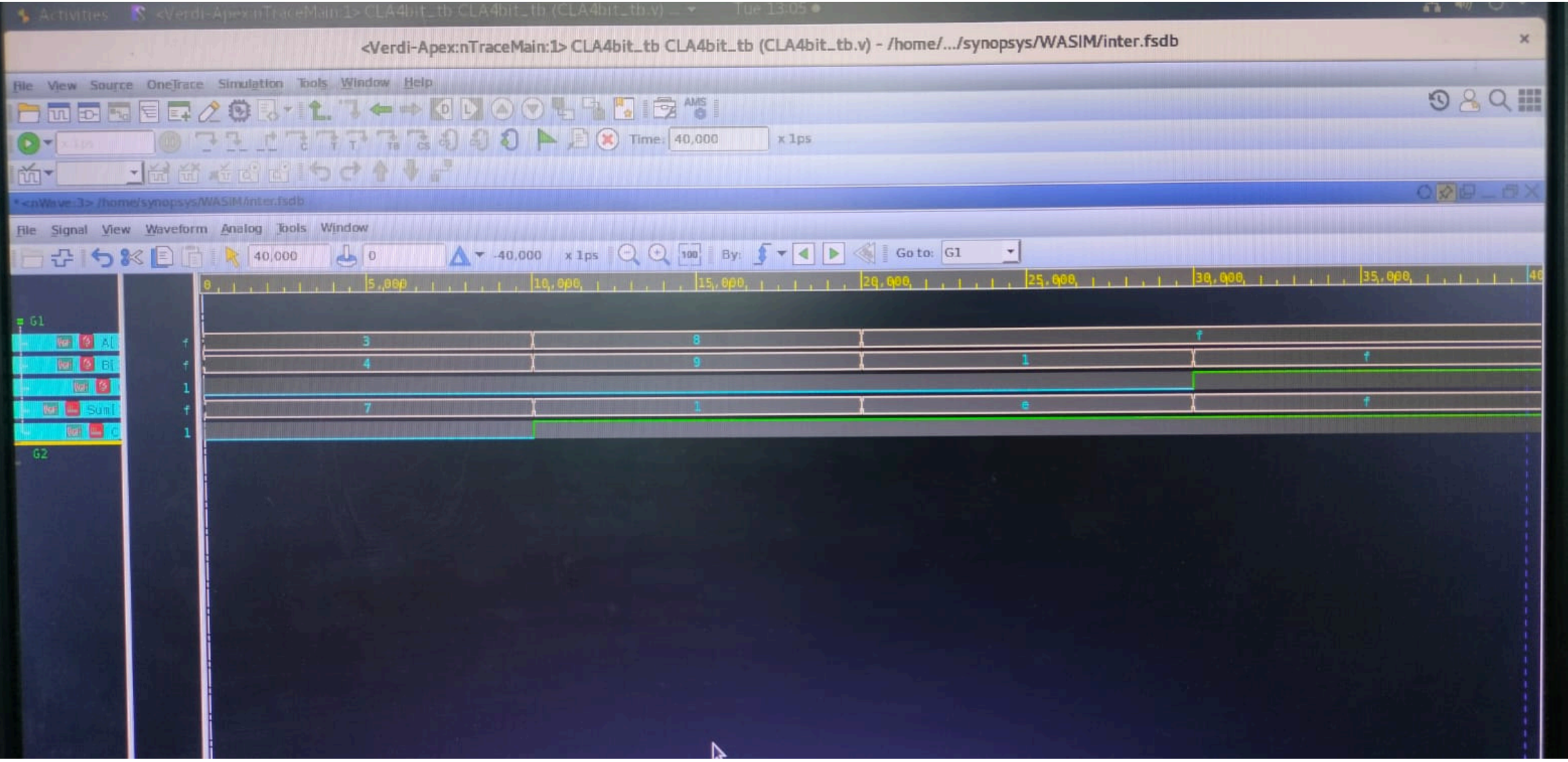
Zoom: Ctrl+Scroll Wheel **Zoom Area:** Ctrl+Drag **Pan:** Middle Click Drag **Push:** Double Click **Cancel Display:** Press 'Escape'

adder4bit_structural.v | CLA_4bit.v | bcd_encoder.srr | bcd_encoder_mult.srs (RTL)[d:0]

Find: Set Filter... Apply Filter Status Page Filter Group

tes (27 filtered)

ID Message Source Location Log Location Time Report


Activities
<Verdi-Apex:nTraceMain:1> CLA4bit_tb CLA4bit_tb (CLA4bit_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 40,000
x 1ps
*<nWave_3> /home/synopsys/WASIM/inter.fsdb
File Signal View Waveform Analog Tools Window
40,000
0
-40,000
x 1ps
By:
Go to: G1
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000
G1
A[
B[
Sum[
C
G2
3
8
f
4
9
1
f
1
7
1
e
f
1

3.2 Verilog Code: 4-bit CLA Testbench: The testbench is identical to the 4-bit RCA testbench, simply instantiating the carry_lookahead_adder.

```
// tb_carry_lookahead_adder.v
`timescale 1ns/1ps

module tb_carry_lookahead_adder;
   reg  [3:0] A, B; reg Cin;
   wire [3:0] Sum; wire Cout;
   carry_lookahead_adder UUT (.*);
   initial begin
      // Test cases are identical to the RCA testbench
      $display("Time\t A    +    B   + Cin = Cout Sum");
      $monitor("%2dns\t %h(h) + %h(h) + %b   =   %b   %h(h)", $time, A, B, Cin, Cout, Sum);
      A = 4'h3; B = 4'h4; Cin = 1'b0; #10; A = 4'h8; B = 4'h9; Cin = 1'b0; #10;
      A = 4'hF; B = 4'h1; Cin = 1'b0; #10; A = 4'hF; B = 4'hF; Cin = 1'b1; #10;
      $finish;
   end
endmodule
```

4. 4-bit Carry-Skip Adder (CSA)

Concept: A compromise between RCA and CLA. It divides the adder into blocks. If an entire block is set to propagate (i.e., P[i]=1 for all bits in the block), the carry-in to the block can skip directly to the next block.

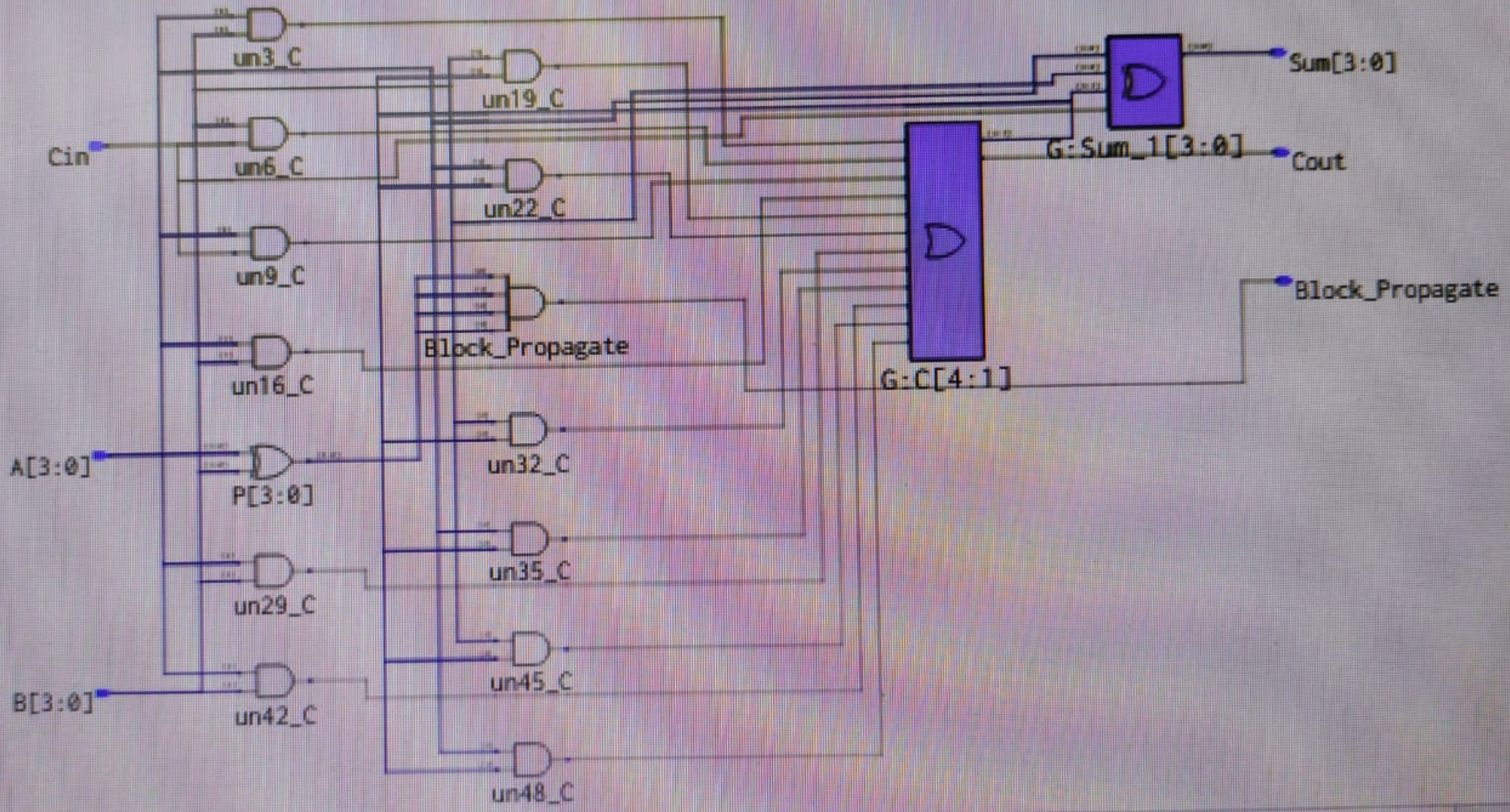
Logic:

- **P_block = P[0] · P[1] · P[2] · P[3] where P[i] = A[i] ⊕ B[i]**
- **Cout = (P_block · Cin) + (!P_block · C4_ripple)**
- **The internal sum uses standard ripple-carry logic.**

4.1 Verilog Code: 4-bit Carry-Skip Adder Design

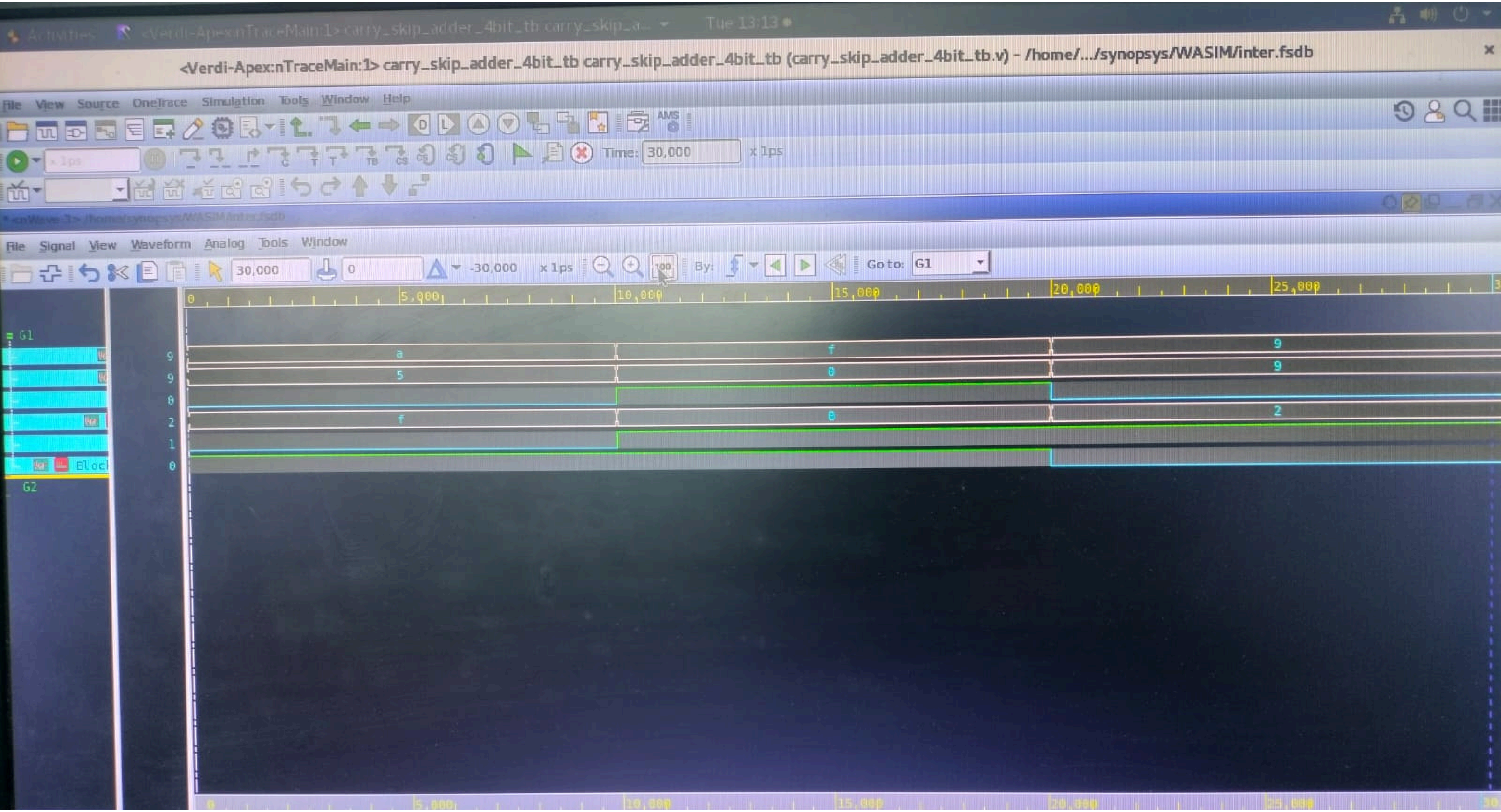
```
// carry_skip_adder_4bit.v
// A single 4-bit block for a Carry-Skip Adder
module carry_skip_adder_4bit (
    input  wire [3:0] A, B,
    input  wire       Cin,
    output wire [3:0] Sum,
    output wire       Cout,
    output wire       Block_Propagate // Signal to
enable skipping
);

    wire [3:0] P;
    wire [4:0] C;

    // Internal ripple-carry logic
    assign C[0] = Cin;
    assign Sum[0] = A[0] ^ B[0] ^ C[0];
    assign C[1] = (A[0] & B[0]) | (B[0] & C[0]) | (A[0]
& C[0]);
    assign Sum[1] = A[1] ^ B[1] ^ C[1];
    assign C[2] = (A[1] & B[1]) | (B[1] & C[1]) | (A[1]
& C[1]);
    assign Sum[2] = A[2] ^ B[2] ^ C[2];
    assign C[3] = (A[2] & B[2]) | (B[2] & C[2]) | (A[2]
& C[2]);
    assign Sum[3] = A[3] ^ B[3] ^ C[3];
    assign C[4] = (A[3] & B[3]) | (B[3] & C[3]) | (A[3]
& C[3]);

    assign Cout = C[4];
```

```
// Calculate individual propagate signals
assign P = A ^ B;

// Calculate the block propagate signal
assign Block_Propagate = P[0] & P[1] & P[2] & P[3];

endmodule
```



Area: Ctrl+Drag **Pan:** Middle Click Drag **Push:** Double Click **Cancel Display:** Press 'Escape'

it.v bcd_encoder_mult.srs (RTL)[d:0]



<Verdi-Apex:nTraceMain:1> carry_skip_adder_4bit_tb carry_skip_adder_4bit_tb (carry_skip_adder_4bit_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 30,000 x 1ps
File Signal View Waveform Analog Tools Window
30,000
0
-30,000
x 1ps
By:
Go to: G1
G1
G2
Block
0
5,000
10,000
15,000
20,000
25,000
a
f
9
5
0
9
f
0
2

Testbench for 4-bit Carry-Skip Adder

```
// tb_carry_skip_adder_4bit.v
`timescale 1ns/1ps

module tb_carry_skip_adder_4bit;

   reg  [3:0] A, B;
   reg       Cin;
   wire [3:0] Sum;
   wire      Cout;
   wire      Block_Propagate;

   carry_skip_adder_4bit UUT (
      .A(A), .B(B), .Cin(Cin),
      .Sum(Sum), .Cout(Cout),
.Block_Propagate(Block_Propagate)
   );

 initial begin
 $display("Time\t A + B + Cin = Cout Sum |
Block_Propagate");
 $monitor("%2dns\t %h + %h + %b = %b %h |
%b", $time, A, B, Cin, Cout, Sum,
Block_Propagate);

 // Case 1: No block propagate
 A = 4'b1010; B = 4'b0101; Cin = 1'b0; #10; //
A^B = 1111, Propagate = 1

 // Case 2: Block propagate is active
 A = 4'b1111; B = 4'b0000; Cin = 1'b1; #10; //
A^B = 1111, Propagate = 1
```

```
// Case 3: No block propagate
A = 4'b1001; B = 4'b1001; Cin = 1'b0; #10; // A^B = 0000, Propagate = 0

$finish;
end

endmodule
```

5. 4-bit Carry-Save Adder

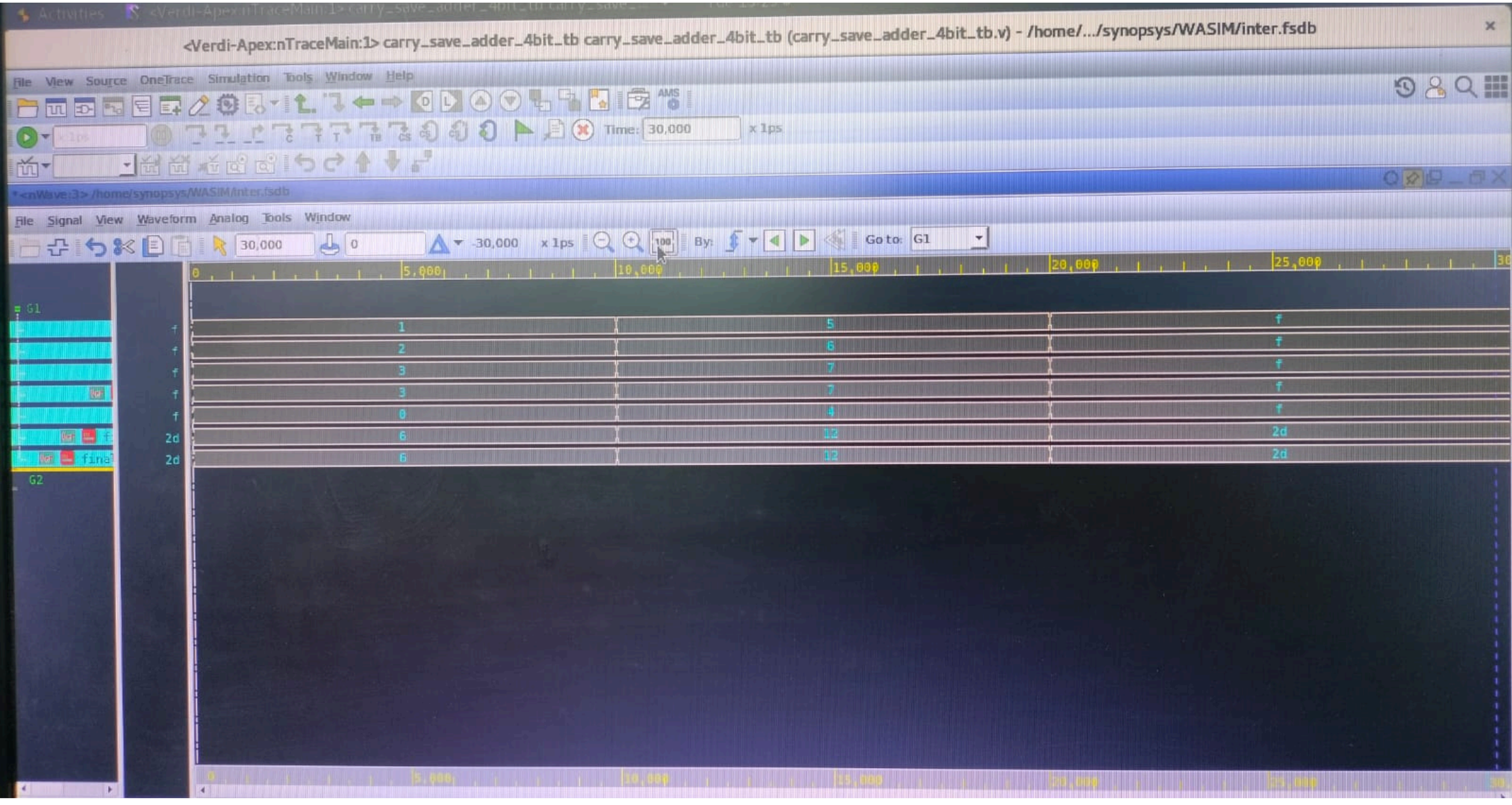
Concept: Used for adding three or more numbers quickly. It does not produce a final sum. Instead, it uses a row of parallel Full Adders with no carry connections between them.

Logic (for each bit i):

- **Sum[i] = A[i] ⊕ B[i] ⊕ C[i]**
- **Carry_Out[i] = (A[i] · B[i]) + (B[i] · C[i]) + (A[i] · C[i])**
- **Final Result = Sum + (Carry_Out << 1)**

5.1 Verilog Code: 4-bit Carry-Save Adder Design:

```
// carry_save_adder_4bit.v
`timescale 1ns/1ps
`include "full_adder_behavioral.v"

module carry_save_adder_4bit (
    input  wire [3:0] A, B, C,
    output wire [3:0] Sum,
    output wire [3:0] Carry_Out
);
    // Four independent Full Adders
    full_adder_behavioral FA0 (A[0], B[0], C[0], Sum[0], Carry_Out[0]);
    full_adder_behavioral FA1 (A[1], B[1], C[1], Sum[1], Carry_Out[1]);
    full_adder_behavioral FA2 (A[2], B[2], C[2], Sum[2], Carry_Out[2]);
    full_adder_behavioral FA3 (A[3], B[3], C[3], Sum[3], Carry_Out[3]);
endmodule
```


Activities
<Verdi-Apex:nTraceMain:1> carry_save_adder_4bit_tb carry_save_adder_4bit_tb (carry_save_adder_4bit_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 30,000
x 1ps
*<nWave:3> /home/synopsys/WASIM/inter.fsdb
File Signal View Waveform Analog Tools Window
30,000
0
-30,000
x 1ps
By:
Go to: G1
0
5,000
10,000
15,000
20,000
25,000
G1
G2
1
5
f
2
6
f
3
7
f
3
7
f
0
4
f
6
12
2d
6
12
2d

5.2 Verilog Code: 4-bit Carry-Save Adder Testbench: The testbench calculates the final sum from the CSA outputs to verify correctness.

```
/ tb_carry_save_adder_4bit.v
`timescale 1ns/1ps

module tb_carry_save_adder_4bit;
 reg [3:0] A, B, C;
 wire [3:0] Sum, Carry_Out;
 wire [7:0] final_sum_csa = Sum + (Carry_Out
<< 1);
 wire [7:0] final_sum_direct = A + B + C;

 carry_save_adder_4bit UUT (.*);

initial begin
 $display("Time\t A+B+C | Sum Carry |
CSA_Res == Direct_Res?");
 $monitor("%2dns\t %h+%h+%h | %h %h | %d
(%h) == %d (%h)",
 $time, A, B, C, Sum, Carry_Out,
 final_sum_csa, final_sum_csa,
final_sum_direct, final_sum_direct);

 A=4'h1; B=4'h2; C=4'h3; #10; // 1+2+3 = 6
 A=4'h5; B=4'h6; C=4'h7; #10; // 5+6+7 = 18
 A=4'hF; B=4'hF; C=4'hF; #10; // 15+15+15 = 45
 $finish;
 end
endmodule
```

6. 4-bit BCD Adder

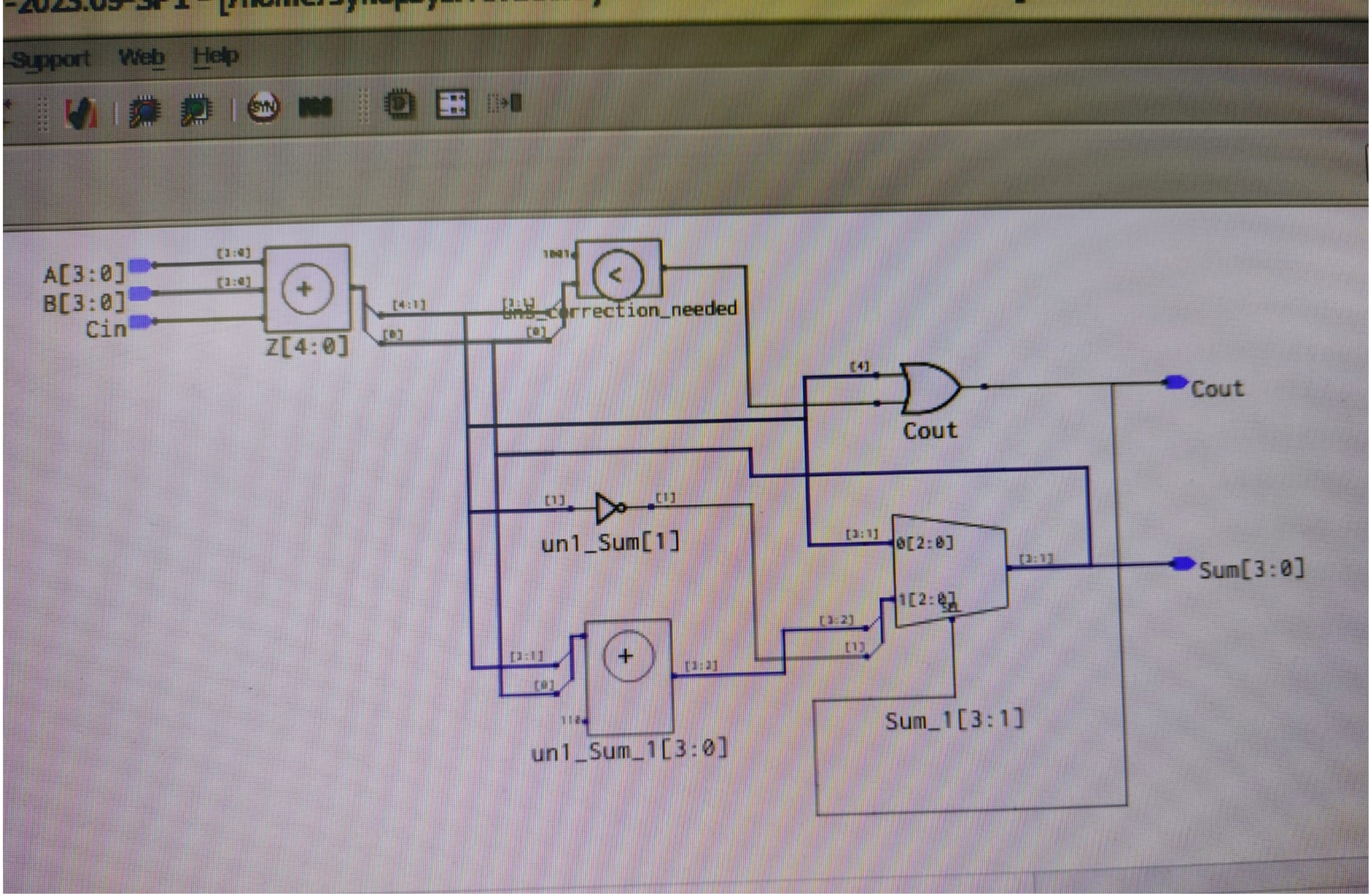
Concept: Adds two 4-bit numbers representing Binary-Coded Decimal digits (0-9). The output must also be a valid BCD digit and a carry.

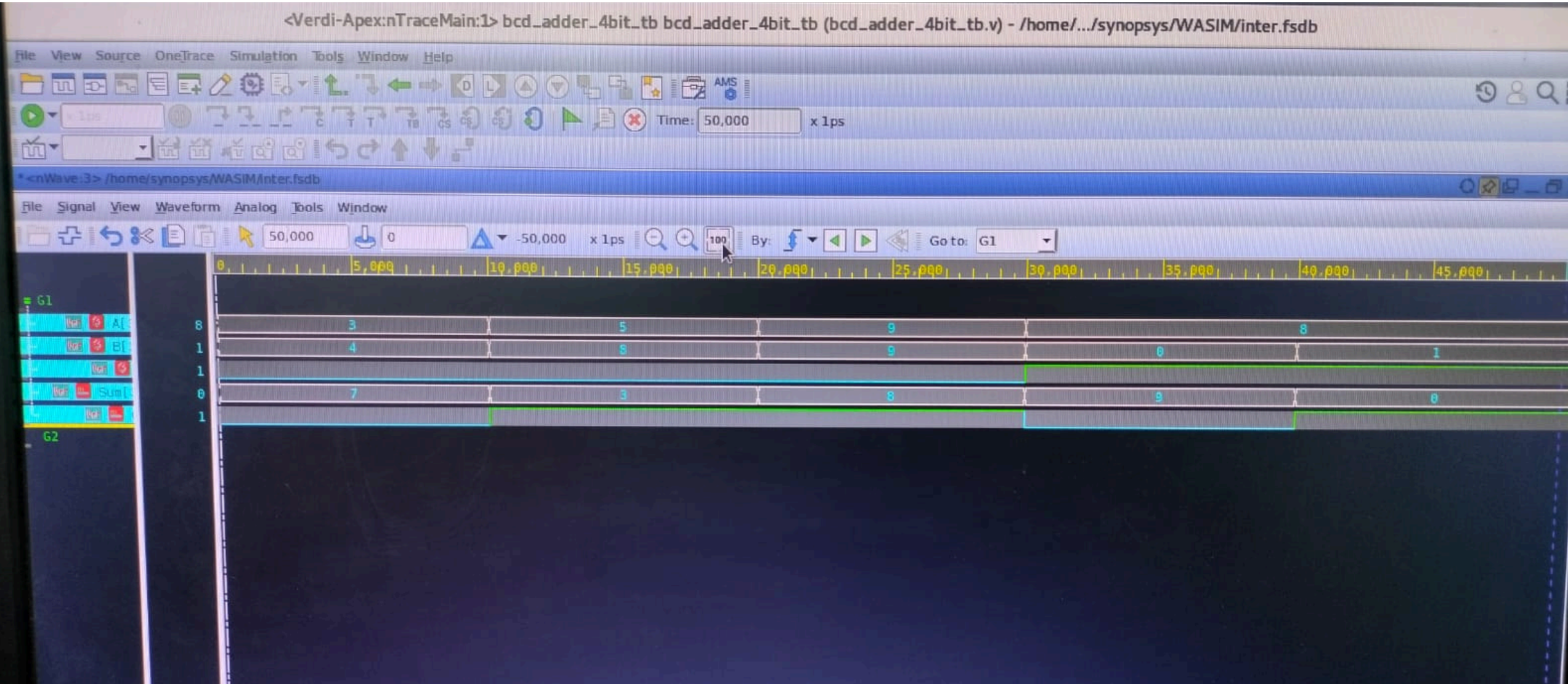
Logic:

1. **Perform a binary addition: Z = A + B + Cin.**
2. **A correction is needed if the binary sum Z is greater than 9. Let Correction = 1 if Z > 9.**
3. **Cout = Correction.**
4. **If Correction = 1, the final Sum = Z + 6. Otherwise, Sum = Z.**

6.1 Verilog Code: 4-bit BCD Adder Design

```
// bcd_adder_4bit.v
// 4-bit BCD Adder
module bcd_adder_4bit (
    input  wire [3:0] A, B,
    input  wire       Cin,
    output wire [3:0] Sum,
    output wire       Cout
);

    wire [4:0] Z; // Intermediate sum
    wire       correction_needed;

    // Step 1: Perform binary addition
    assign Z = A + B + Cin;

    // Step 2: Check if correction is needed
    // Correction is needed if the binary sum is > 9,
or if the intermediate carry is 1.
    assign correction_needed = (Z[3:0] > 9) || Z[4];

    // Step 3: Perform correction if needed
    // If correction is needed, add 6 (0110).
    // The final carry is 1 if correction was needed.
    assign Cout = correction_needed;
    assign Sum = correction_needed ? Z[3:0] +
4'd6 : Z[3:0];

endmodule
```


Support
Web
Help
A[3:0]
B[3:0]
Cin
Z[4:0]
un1_Sum[1]
un1_Sum_1[3:0]
Sum_1[3:1]
Cout
Cout
Sum[3:0]


<Verdi-Apex:nTraceMain:1> bcd_adder_4bit_tb bcd_adder_4bit_tb (bcd_adder_4bit_tb.v) - /home/.../synopsys/WASIM/inter.fsdb
File View Source OneTrace Simulation Tools Window Help
Time: 50,000
x 1ps
*<nWave:3> /home/synopsys/WASIM/inter.fsdb
File Signal View Waveform Analog Tools Window
50,000
0
-50,000
x 1ps
By:
Go to: G1
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000
40,000
45,000
G1
A[
B[
Sum[
G2
8
1
1
0
1
3
5
9
8
4
8
9
0
1
7
3
8
9
0

6.2 Verilog Code: 4-bit BCD Adder Testbench

```
// tb_bcd_adder_4bit.v
`timescale 1ns/1ps

module tb_bcd_adder_4bit;

    reg  [3:0] A, B;
    reg        Cin;
    wire [3:0] Sum;
    wire       Cout;

    bcd_adder_4bit UUT (
        .A(A), .B(B), .Cin(Cin),
        .Sum(Sum), .Cout(Cout)
    );

    initial begin
        $display("Time\t A + B + Cin = BCD_Cout BCD_Sum");
        $monitor("%2dns\t %d + %d + %b = %b %d", $time, A, B, Cin, Cout, Sum);

        // Case 1: No correction (3 + 4 = 7)
        A = 4'd3; B = 4'd4; Cin = 1'b0; #10;

        // Case 2: Correction needed because sum > 9 (5 + 8 = 13 -> 1 0011)
        A = 4'd5; B = 4'd8; Cin = 1'b0; #10;
```

```
// Case 3: Correction needed because of intermediate carry (9 + 9 = 18 -> 1 1000)
A = 4'd9; B = 4'd9; Cin = 1'b0; #10;

// Case 4: With carry-in (8 + 0 + 1 = 9)
A = 4'd8; B = 4'd0; Cin = 1'b1; #10;

// Case 5: With carry-in causing correction (8 + 1 + 1 = 10 -> 1 0000)
A = 4'd8; B = 4'd1; Cin = 1'b1; #10;

$finish;
end

endmodule
```