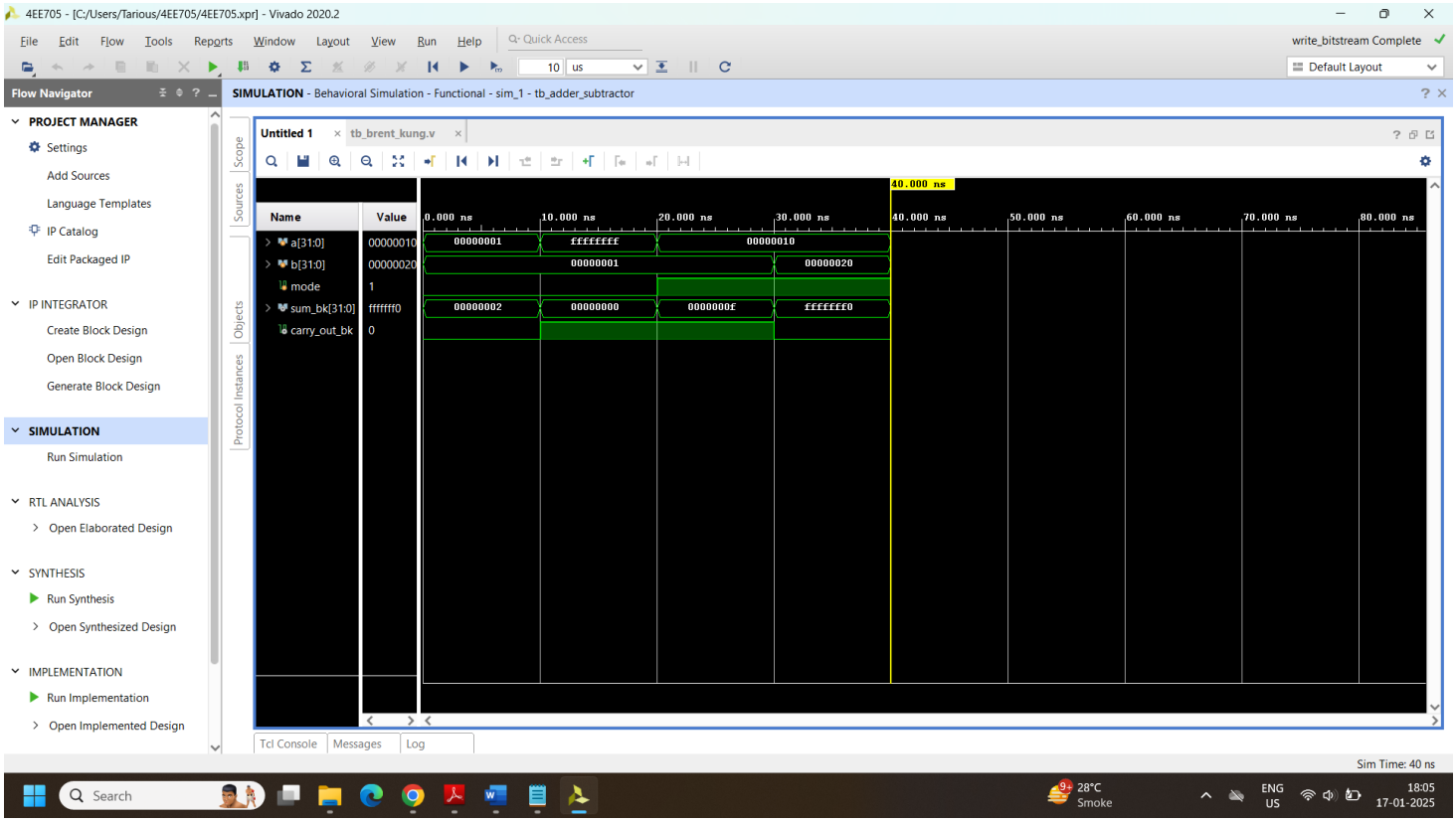
Design a 32-bit Brent-Kung adder. For testing the adder, you will be using BRAM, VIO, ILA and Control unit (which can read the data from BRAM and send it to your ADDER module).

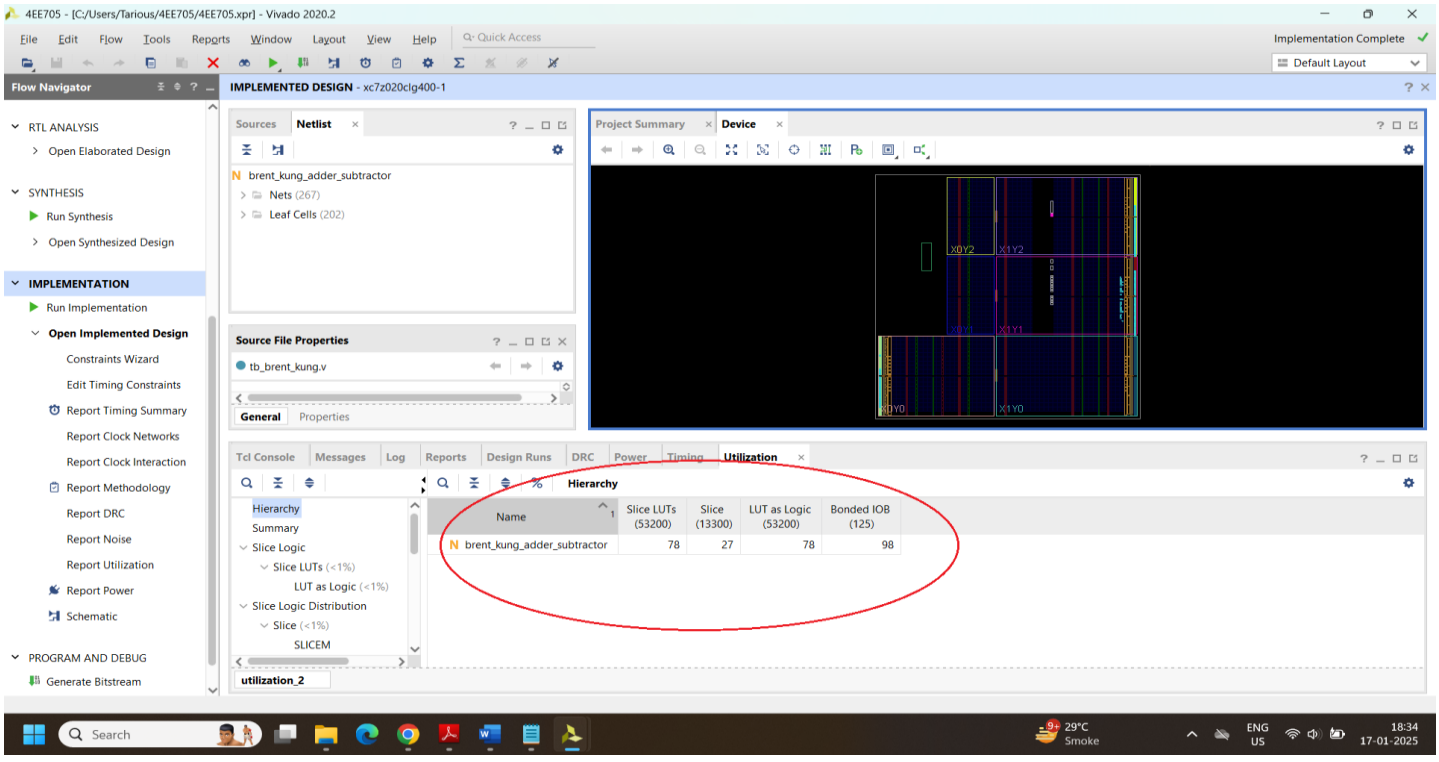
Task-1 - Design a 32-bit Adder/Subtractor and write a testbench and simulate the circuit.

Design Using Brent-Kung Adder

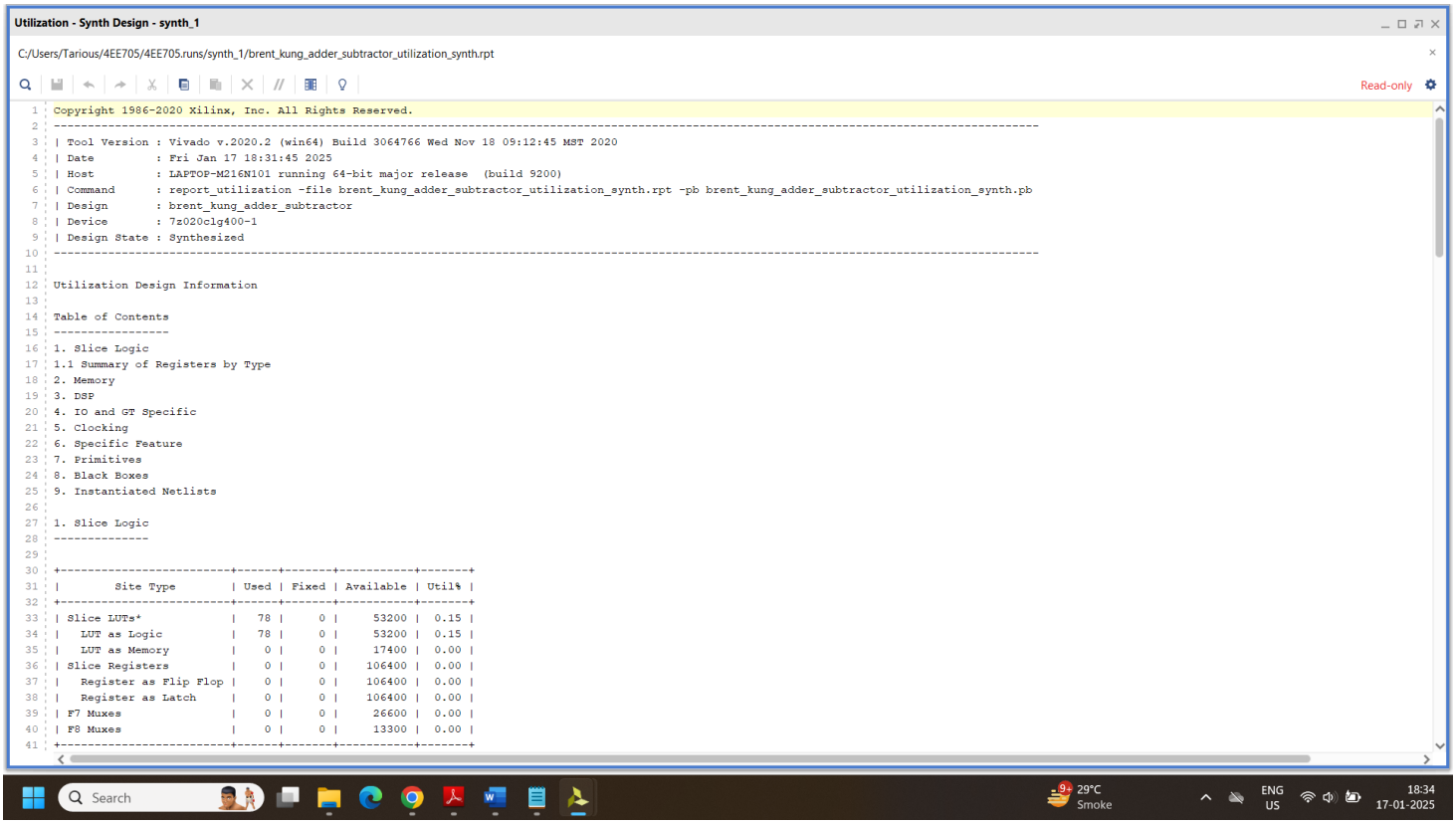
Simulation Waveform Using Brent Kung Adder



Utilisation:



Resource Utilisation:



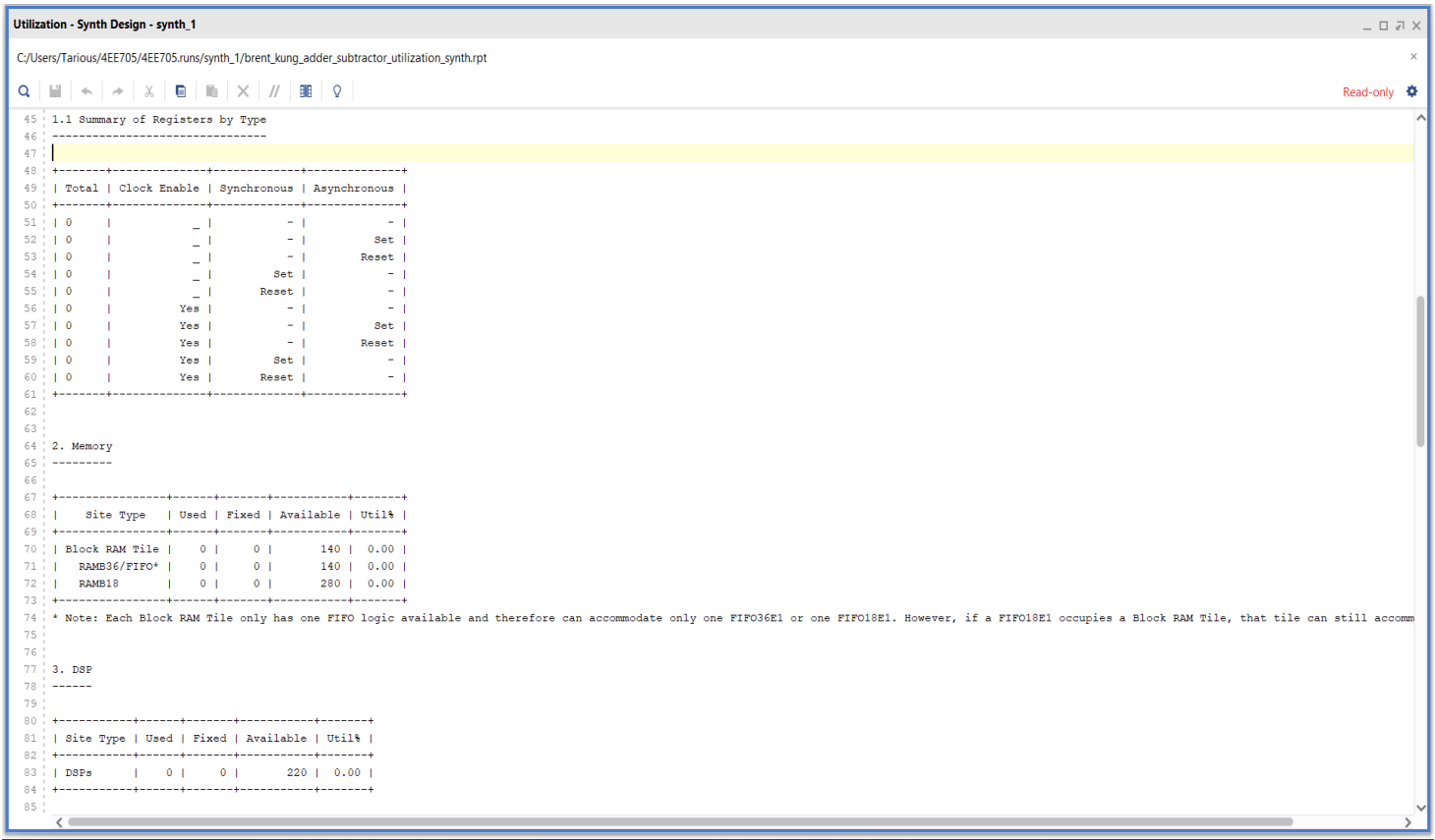
Utilization - Synth Design - synth_1

C:/Users/Tarious/4EE705/4EE705.runs/synth_1/brent_kung_adder_subtractor_utilization_synth.rpt

Read-only

```
Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.
----------------------------------------------------------------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2020.2 (win64) Build 3064766 Wed Nov 18 09:12:45 MST 2020
| Date         : Fri Jan 17 18:31:45 2025
| Host         : LAPTOP-M216N101 running 64-bit major release  (build 9200)
| Command      : report_utilization -file brent_kung_adder_subtractor_utilization_synth.rpt -pb brent_kung_adder_subtractor_utilization_synth.pb
| Design       : brent_kung_adder_subtractor
| Device       : 7z020clg400-1
| Design State : Synthesized
----------------------------------------------------------------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |   78 |     0 |     53200 |  0.15 |
|   LUT as Logic          |   78 |     0 |     53200 |  0.15 |
|   LUT as Memory         |    0 |     0 |     17400 |  0.00 |
| Slice Registers         |    0 |     0 |    106400 |  0.00 |
|   Register as Flip Flop |    0 |     0 |    106400 |  0.00 |
|   Register as Latch     |    0 |     0 |    106400 |  0.00 |
| F7 Muxes                |    0 |     0 |     26600 |  0.00 |
| F8 Muxes                |    0 |     0 |     13300 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```


Search 29°C Smoke ENG US 18:34 17-01-2025

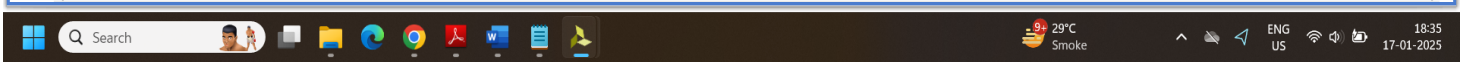


Utilization - Synth Design - synth_1

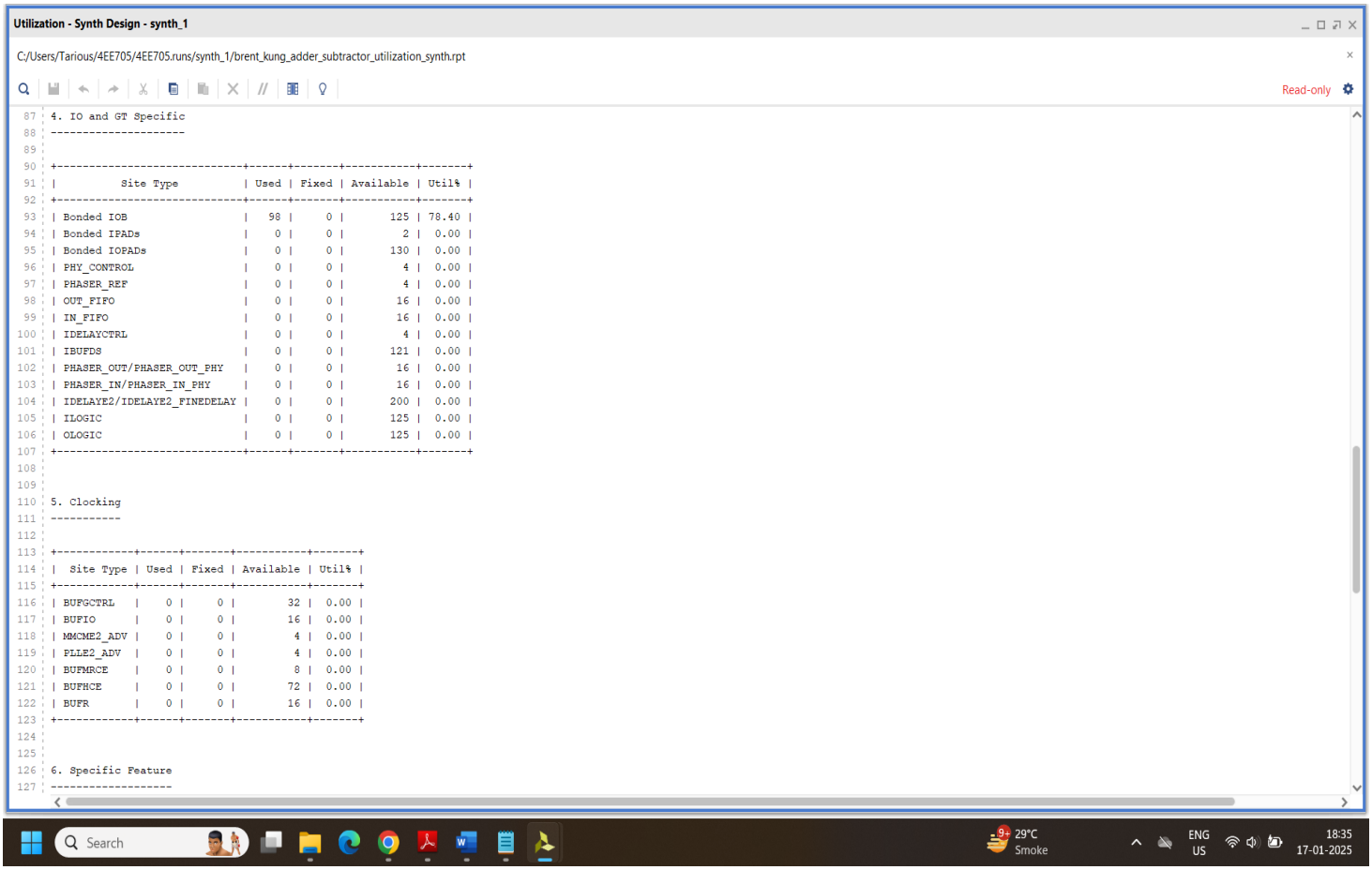
C:/Users/Tarious/4EE705/4EE705.runs/synth_1/brent_kung_adder_subtractor_utilization_synth.rpt

Read-only

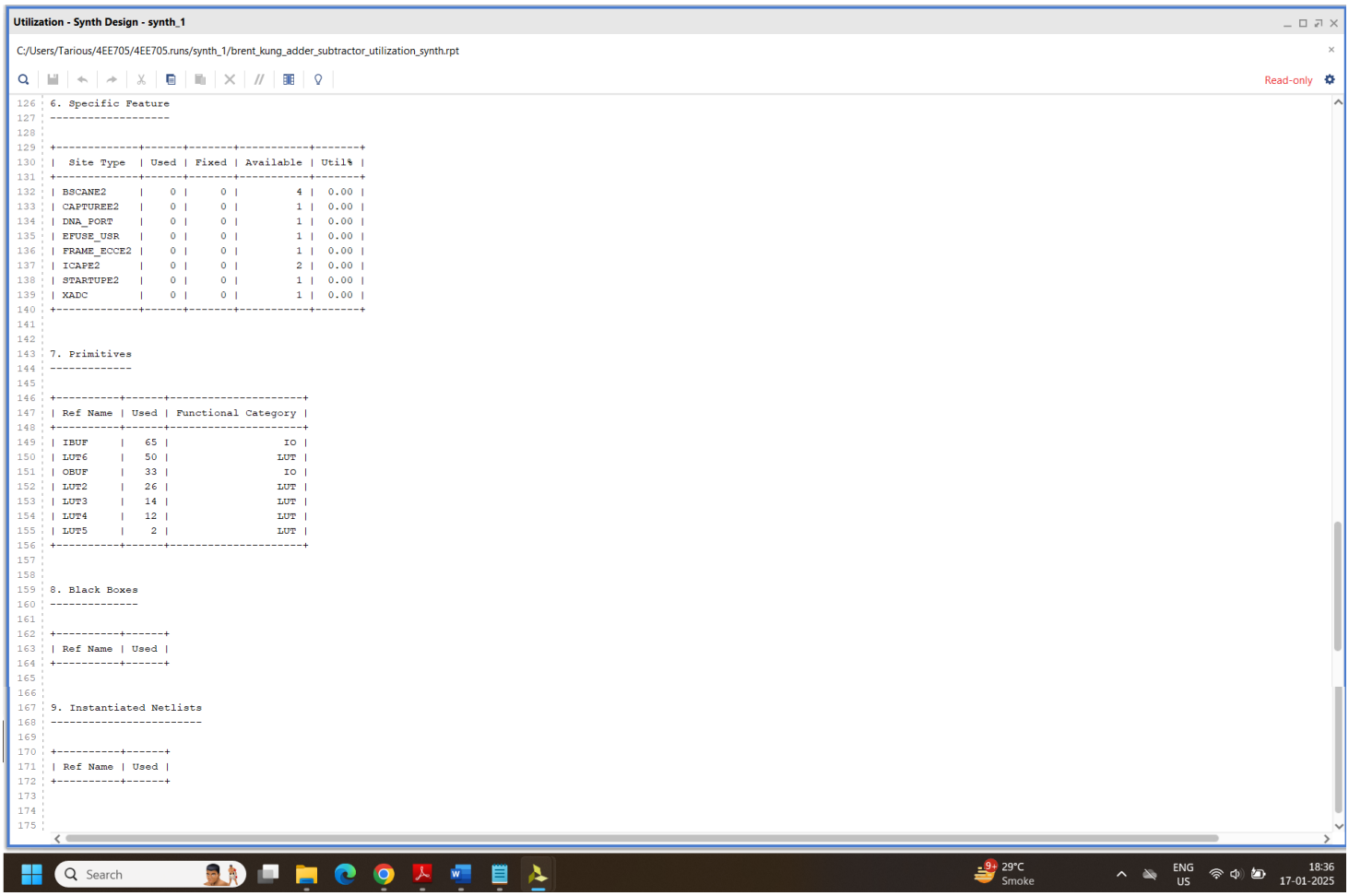
```
1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 0     |          Yes |           - |          Set |
| 0     |          Yes |           - |        Reset |
| 0     |          Yes |         Set |            - |
| 0     |          Yes |       Reset |            - |
+-------+--------------+-------------+--------------+


2. Memory
---------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| Block RAM Tile |    0 |     0 |       140 |  0.00 |
|   RAMB36/FIFO* |    0 |     0 |       140 |  0.00 |
|   RAMB18       |    0 |     0 |       280 |  0.00 |
+----------------+------+-------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accomm


3. DSP
------

+-----------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+-----------+------+-------+-----------+-------+
| DSPs      |    0 |     0 |       220 |  0.00 |
+-----------+------+-------+-----------+-------+
```



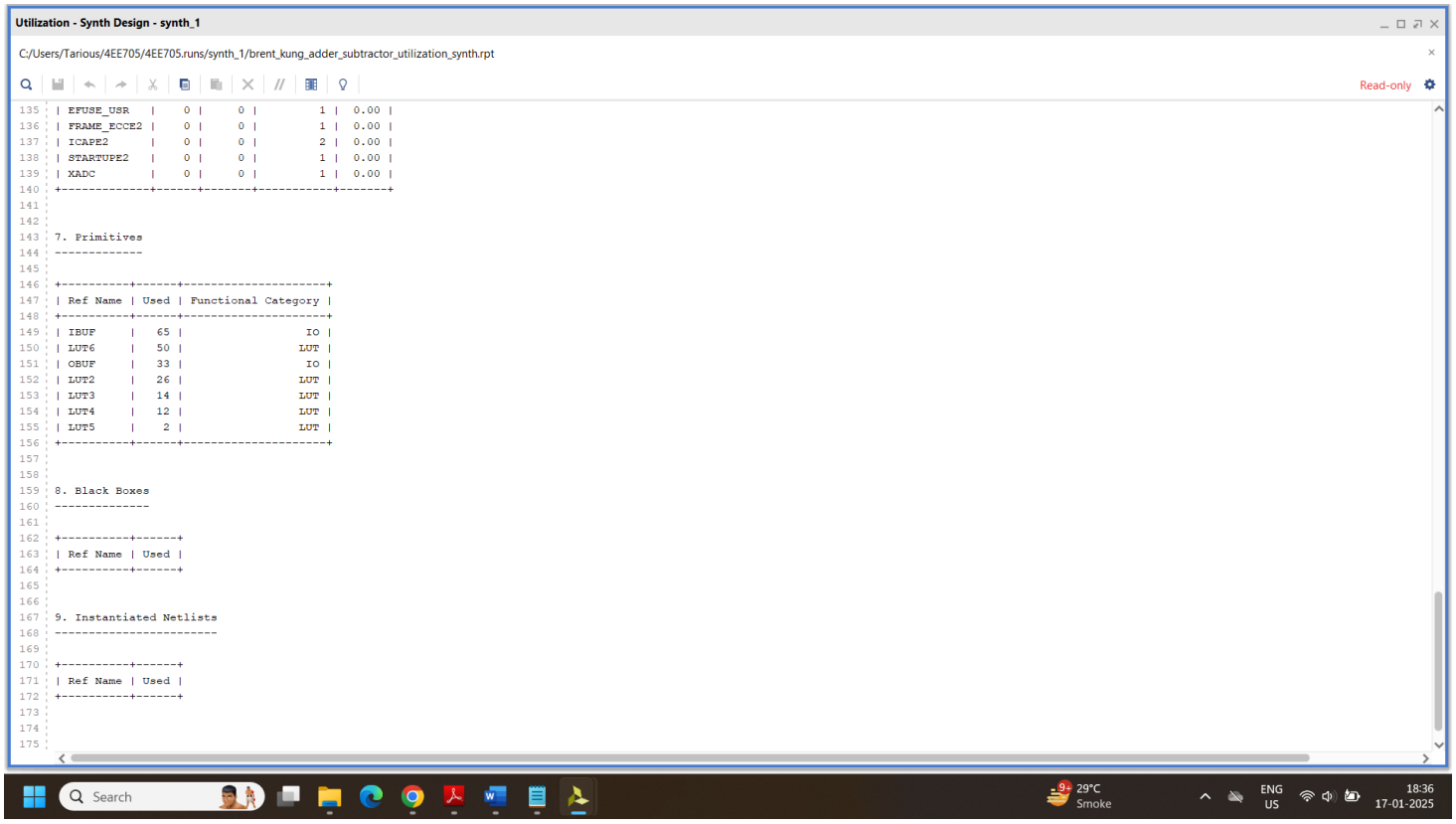
Search 29°C Smoke ENG US 18:35 17-01-2025




```
4. IO and GT Specific
---------------------

+-------------------------------+------+-------+-----------+-------+
|          Site Type            | Used | Fixed | Available | Util% |
+-------------------------------+------+-------+-----------+-------+
| Bonded IOB                    |   98 |     0 |       125 | 78.40 |
| Bonded IPADs                  |    0 |     0 |         2 |  0.00 |
| Bonded IOPADs                 |    0 |     0 |       130 |  0.00 |
| PHY_CONTROL                   |    0 |     0 |         4 |  0.00 |
| PHASER_REF                    |    0 |     0 |         4 |  0.00 |
| OUT_FIFO                      |    0 |     0 |        16 |  0.00 |
| IN_FIFO                       |    0 |     0 |        16 |  0.00 |
| IDELAYCTRL                    |    0 |     0 |         4 |  0.00 |
| IBUFDS                        |    0 |     0 |       121 |  0.00 |
| PHASER_OUT/PHASER_OUT_PHY     |    0 |     0 |        16 |  0.00 |
| PHASER_IN/PHASER_IN_PHY       |    0 |     0 |        16 |  0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY   |    0 |     0 |       200 |  0.00 |
| ILOGIC                        |    0 |     0 |       125 |  0.00 |
| OLOGIC                        |    0 |     0 |       125 |  0.00 |
+-------------------------------+------+-------+-----------+-------+


5. Clocking
-----------

+------------+------+-------+-----------+-------+
|  Site Type | Used | Fixed | Available | Util% |
+------------+------+-------+-----------+-------+
| BUFGCTRL   |    0 |     0 |        32 |  0.00 |
| BUFIO      |    0 |     0 |        16 |  0.00 |
| MMCME2_ADV |    0 |     0 |         4 |  0.00 |
| PLLE2_ADV  |    0 |     0 |         4 |  0.00 |
| BUFMRCE    |    0 |     0 |         8 |  0.00 |
| BUFHCE     |    0 |     0 |        72 |  0.00 |
| BUFR       |    0 |     0 |        16 |  0.00 |
+------------+------+-------+-----------+-------+


6. Specific Feature
-------------------
```


```
6. Specific Feature
-------------------

+-------------+------+-------+-----------+-------+
|  Site Type  | Used | Fixed | Available | Util% |
+-------------+------+-------+-----------+-------+
| BSCANE2     |    0 |     0 |         4 |  0.00 |
| CAPTUREE2   |    0 |     0 |         1 |  0.00 |
| DNA_PORT    |    0 |     0 |         1 |  0.00 |
| EFUSE_USR   |    0 |     0 |         1 |  0.00 |
| FRAME_ECCE2 |    0 |     0 |         1 |  0.00 |
| ICAPE2      |    0 |     0 |         2 |  0.00 |
| STARTUPE2   |    0 |     0 |         1 |  0.00 |
| XADC        |    0 |     0 |         1 |  0.00 |
+-------------+------+-------+-----------+-------+


7. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| IBUF     |   65 |                  IO |
| LUT6     |   50 |                 LUT |
| OBUF     |   33 |                  IO |
| LUT2     |   26 |                 LUT |
| LUT3     |   14 |                 LUT |
| LUT4     |   12 |                 LUT |
| LUT5     |    2 |                 LUT |
+----------+------+---------------------+


8. Black Boxes
--------------

+----------+------+
| Ref Name | Used |
+----------+------+


9. Instantiated Netlists
------------------------

+----------+------+
| Ref Name | Used |
+----------+------+
```



Verilog Code:

// 32-bit Brent-Kung Adder/Subtractor Implementation

```
module brent_kung_adder_subtractor(
    input [31:0] a,
    input [31:0] b,
    input mode,  // mode = 0 for addition, mode = 1 for subtraction
    output [31:0] sum,
    output carry_out
);
    wire [31:0] b_xor_mode;
    wire carry_in;
    assign b_xor_mode = b ^ {32{mode}}; // XOR for 2's complement subtraction
    assign carry_in = mode;
    wire [31:0] p, g, c;
    // Generate propagate (p) and generate (g) signals
    assign p = a ^ b_xor_mode;
    assign g = a & b_xor_mode;
```

```
    // Brent-Kung Prefix Tree for Carry Generation
    genvar i;
    generate
        for (i = 0; i < 32; i = i + 1) begin
            if (i == 0) assign c[i] = g[i] | (p[i] & carry_in);
            else assign c[i] = g[i] | (p[i] & c[i-1]);
        end
    endgenerate

    // Compute Sum
    assign sum = p ^ {c[30:0], carry_in};
    assign carry_out = c[31];
endmodule
```

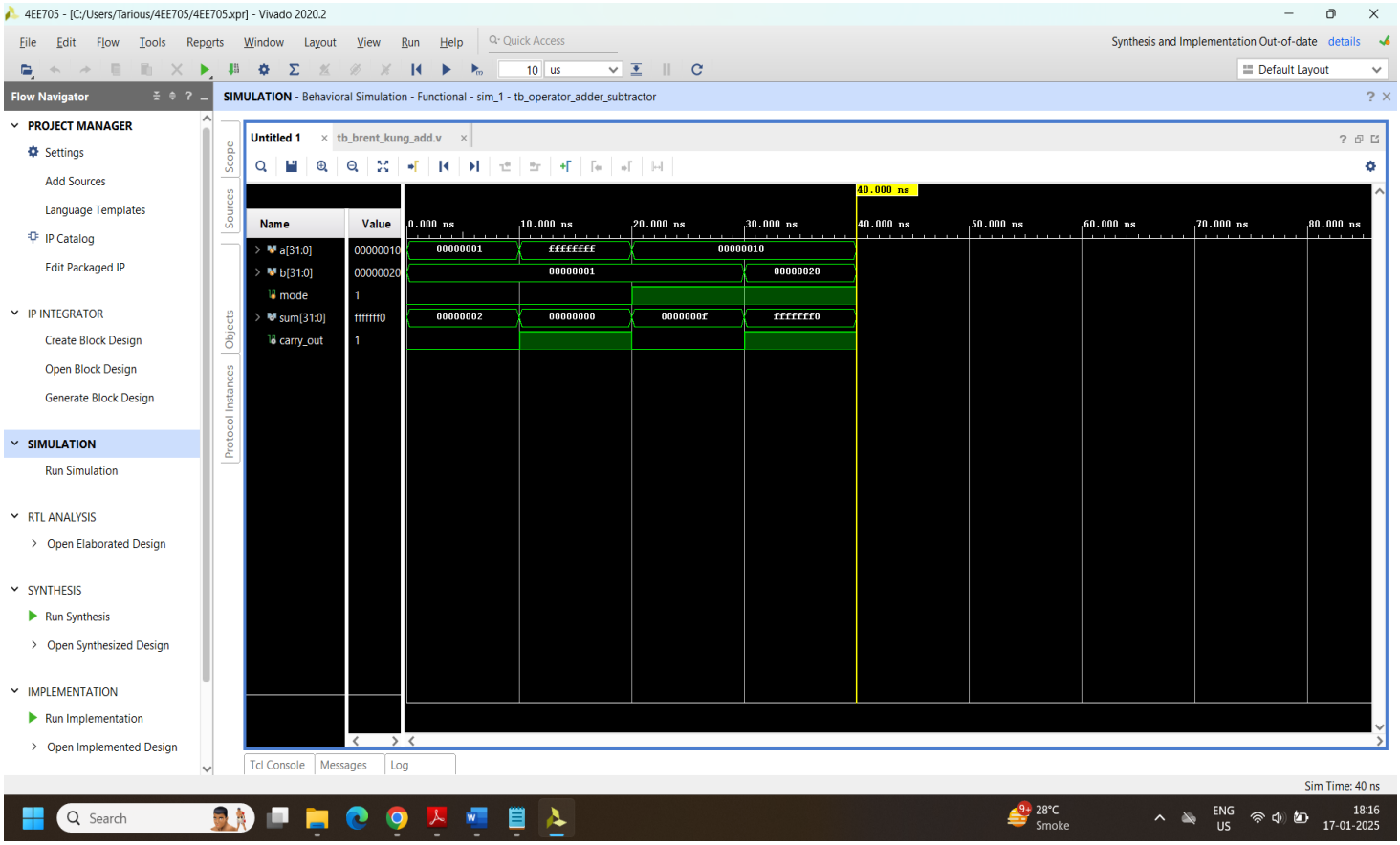
Test Bench Code:

```
// Testbench for 32-bit Brent-Kung Adder/Subtractor
module tb_adder_subtractor;
    reg [31:0] a, b;
    reg mode;
    wire [31:0] sum_bk;
    wire carry_out_bk;
    // Instantiate Module
    brent_kung_adder_subtractor uut1 (
        .a(a),
        .b(b),
        .mode(mode),
        .sum(sum_bk),
        .carry_out(carry_out_bk)
    );
```

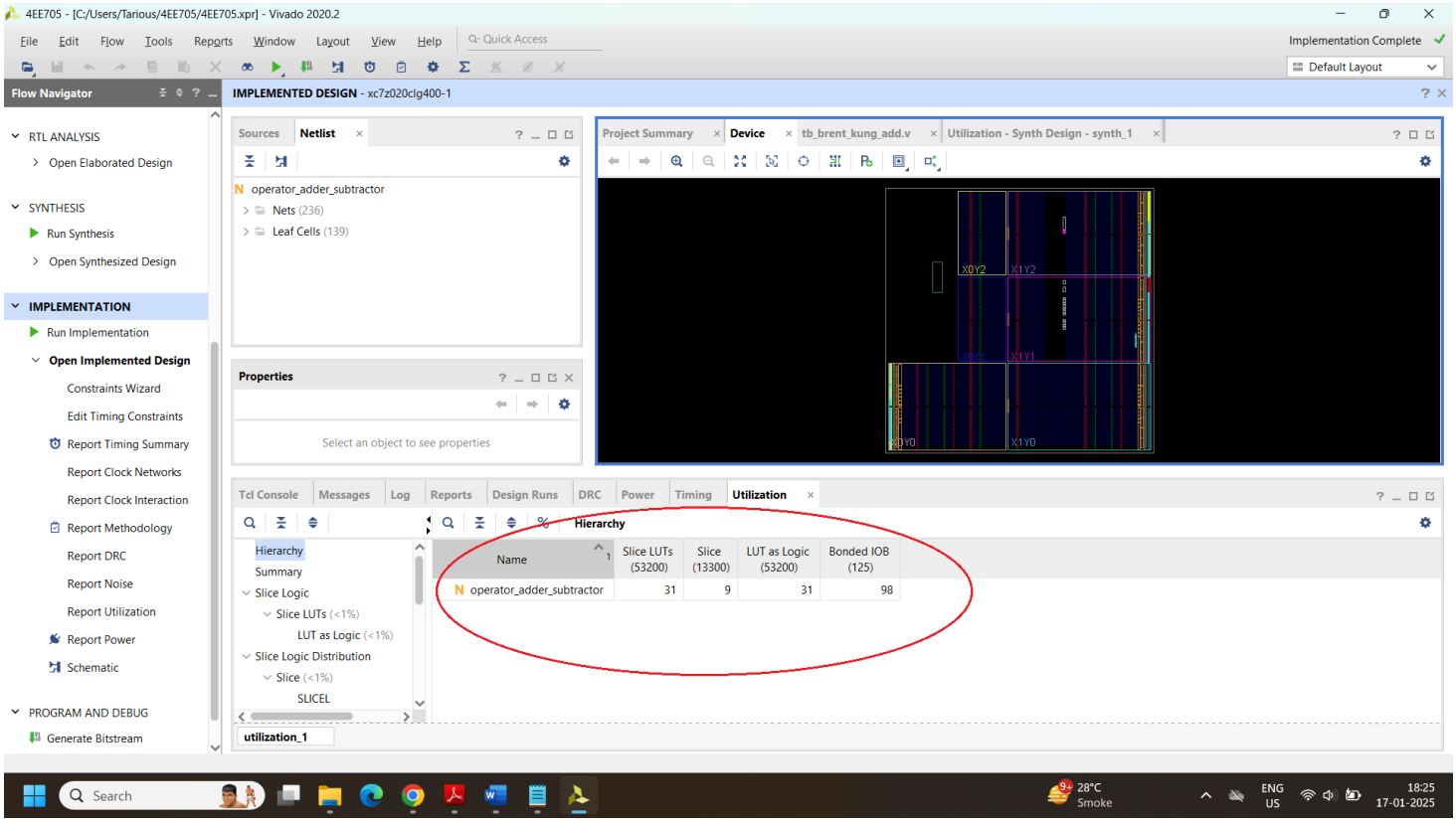
```
initial begin
    // Test Cases
    $display("Time\tMode\tA\tB\tSum_BK\tCarry_BK");
    $monitor("%0t\t%b\t%h\t%h\t%h\t%b", $time, mode, a, b, sum_bk, carry_out_bk);

    a = 32'h00000001; b = 32'h00000001; mode = 0; #10; // Addition
    a = 32'hFFFFFFFF; b = 32'h00000001; mode = 0; #10; // Addition with overflow
    a = 32'h00000010; b = 32'h00000001; mode = 1; #10; // Subtraction
    a = 32'h00000010; b = 32'h00000020; mode = 1; #10; // Subtraction with negative result

    $finish;
  end
endmodule
```

Using + operator only

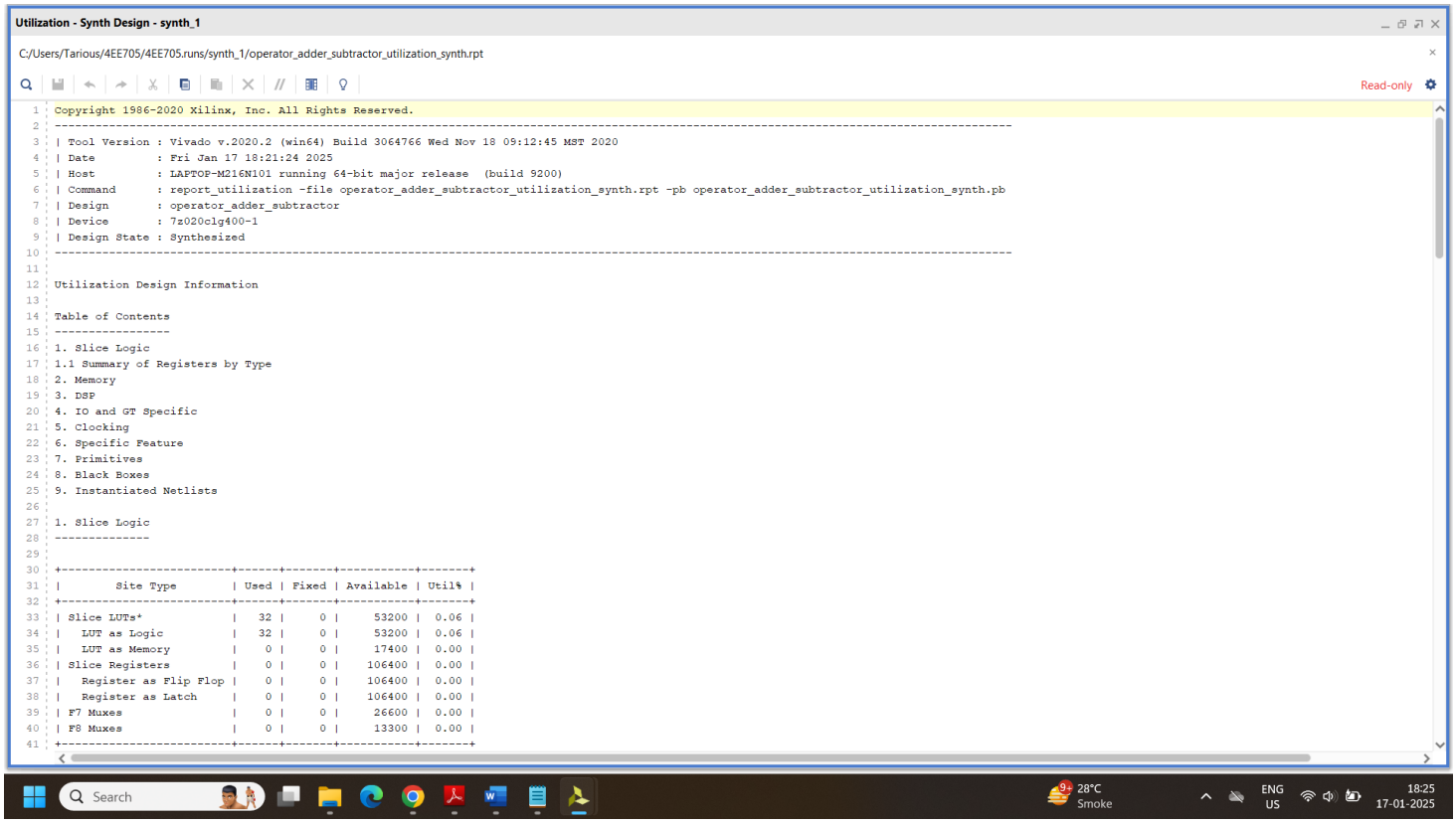
Simulation Waveform Using + Operator Only

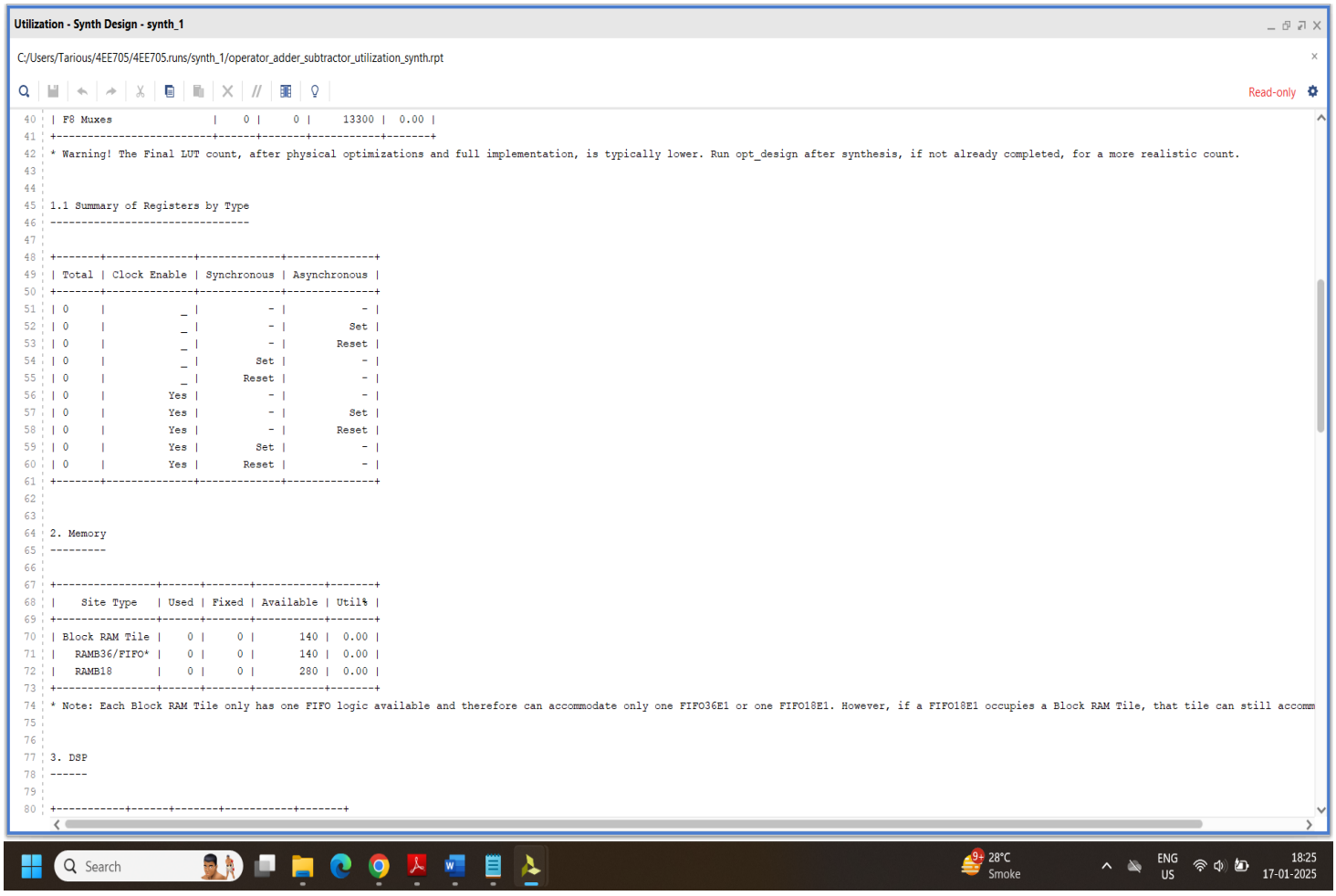


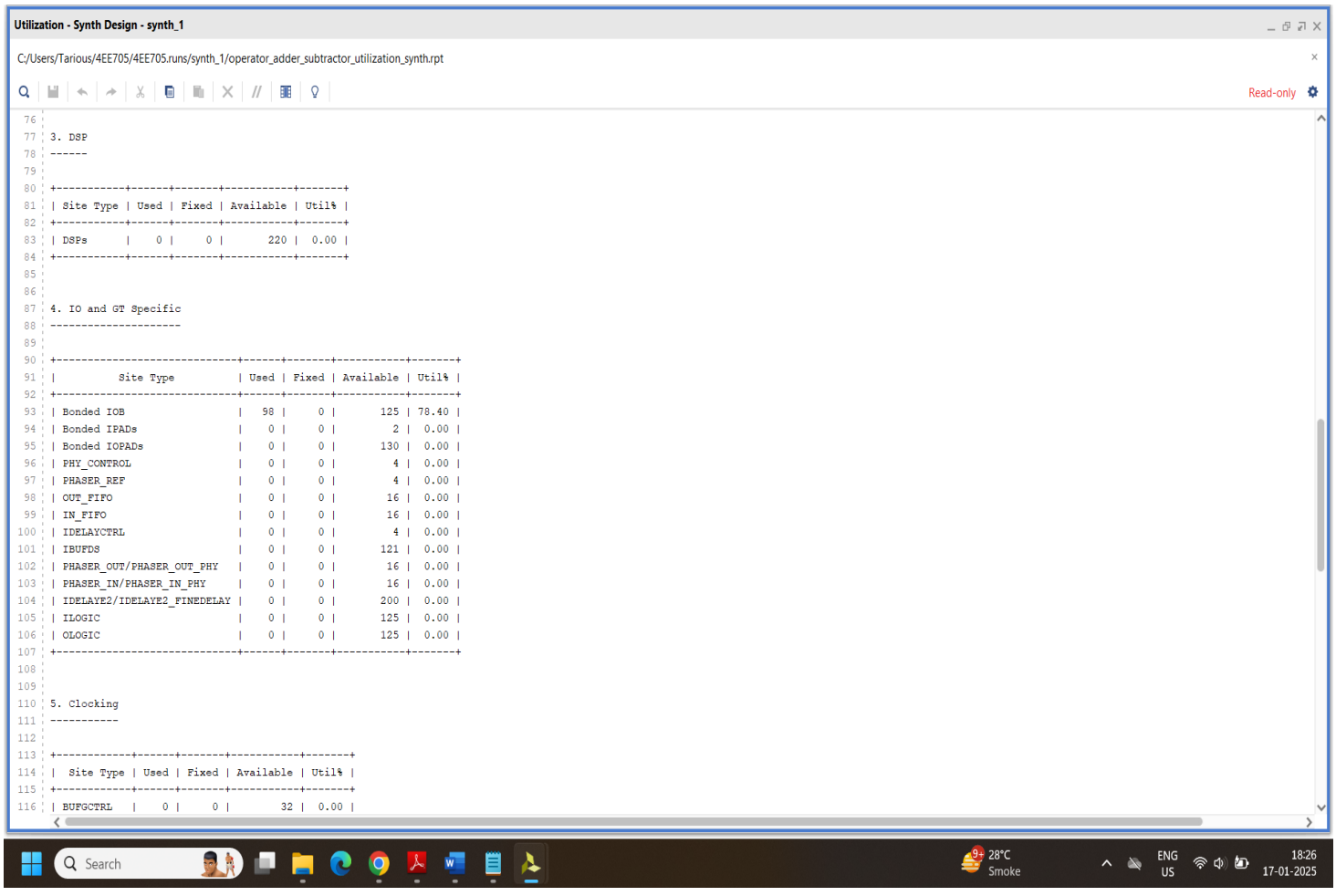
Utilisation:

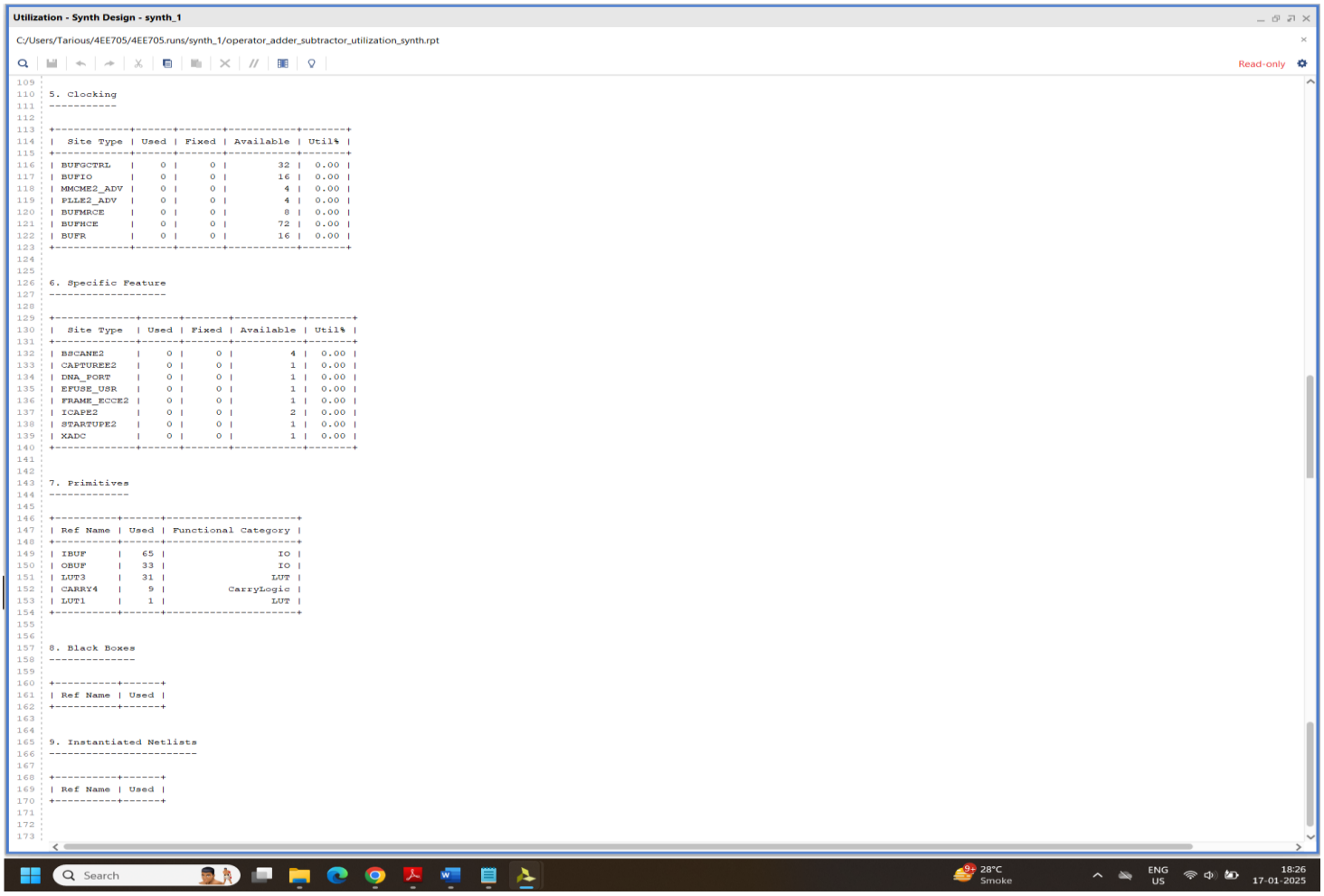


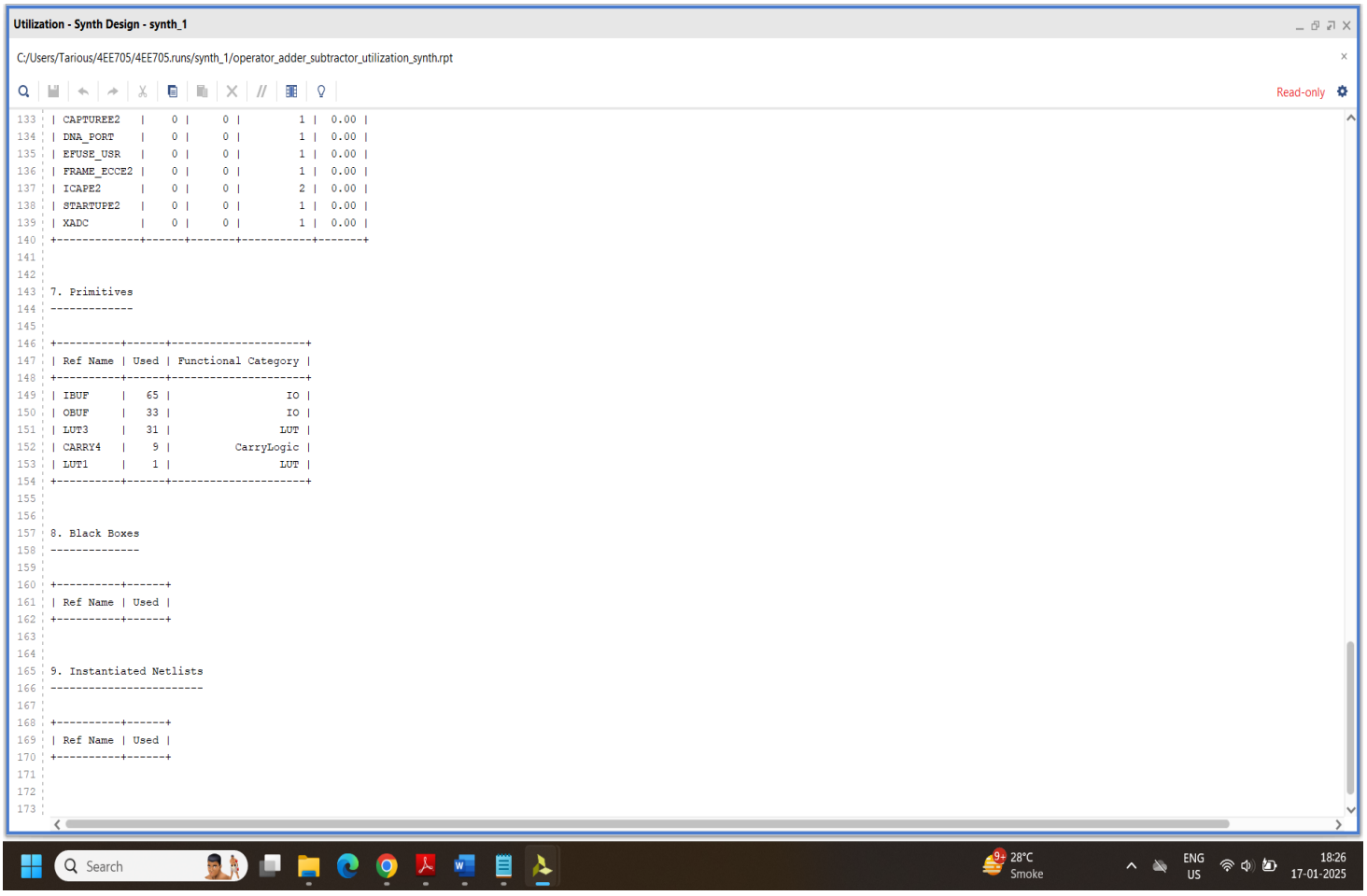
Resource Utilisation:




Utilization - Synth Design - synth_1
C:/Users/Tarious/4EE705/4EE705.runs/synth_1/operator_adder_subtractor_utilization_synth.rpt
Read-only
| F8 Muxes | 0 | 0 | 13300 | 0.00 |
* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more realistic count.
1.1 Summary of Registers by Type
Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
0	Yes	-	Reset
0	Yes	Set	-
0	Yes	Reset	-
2. Memory
Site Type	Used	Fixed	Available	Util%
Block RAM Tile	0	0	140	0.00
RAMB36/FIFO*	0	0	140	0.00
RAMB18	0	0	280	0.00
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accomm
3. DSP
Search
28°C Smoke
ENG US
18:25 17-01-2025


Utilization - Synth Design - synth_1
C:/Users/Tarious/4EE705/4EE705.runs/synth_1/operator_adder_subtractor_utilization_synth.rpt
Read-only
3. DSP
| Site Type | Used | Fixed | Available | Util% |
| DSPs | 0 | 0 | 220 | 0.00 |
4. IO and GT Specific
Site Type	Used	Fixed	Available	Util%
Bonded IOB	98	0	125	78.40
Bonded IPADs	0	0	2	0.00
Bonded IOPADs	0	0	130	0.00
PHY_CONTROL	0	0	4	0.00
PHASER_REF	0	0	4	0.00
OUT_FIFO	0	0	16	0.00
IN_FIFO	0	0	16	0.00
IDELAYCTRL	0	0	4	0.00
IBUFDS	0	0	121	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	16	0.00
PHASER_IN/PHASER_IN_PHY	0	0	16	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	200	0.00
ILOGIC	0	0	125	0.00
OLOGIC	0	0	125	0.00
5. Clocking
Site Type	Used	Fixed	Available	Util%
BUFGCTRL	0	0	32	0.00
Search
28°C Smoke
ENG US
18:26 17-01-2025


Utilization - Synth Design - synth_1
C:/Users/Tarious/4EE705/4EE705.runs/synth_1/operator_adder_subtractor_utilization_synth.rpt
Read-only
5. Clocking
+------------+------+-------+------------+-------+
Site Type	Used	Fixed	Available	Util%
BUFGCTRL	0	0	32	0.00
BUFIO	0	0	16	0.00
MMCME2_ADV	0	0	4	0.00
PLLE2_ADV	0	0	4	0.00
BUFMRCE	0	0	8	0.00
BUFHCE	0	0	72	0.00
BUFR	0	0	16	0.00
6. Specific Feature
Site Type	Used	Fixed	Available	Util%
BSCANE2	0	0	4	0.00
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
STARTUPE2	0	0	1	0.00
XADC	0	0	1	0.00
7. Primitives
Ref Name	Used	Functional Category
IBUF	65	IO
OBUF	33	IO
LUT3	31	LUT
CARRY4	9	CarryLogic
LUT1	1	LUT
8. Black Boxes
Ref Name	Used
9. Instantiated Netlists
Ref Name	Used
Search
28°C
Smoke
ENG
US
18:26
17-01-2025


Utilization - Synth Design - synth_1
C:/Users/Tarious/4EE705/4EE705.runs/synth_1/operator_adder_subtractor_utilization_synth.rpt
Read-only
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
STARTUPE2	0	0	1	0.00
XADC	0	0	1	0.00
7. Primitives
Ref Name	Used	Functional Category
IBUF	65	IO
OBUF	33	IO
LUT3	31	LUT
CARRY4	9	CarryLogic
LUT1	1	LUT
8. Black Boxes
Ref Name	Used
9. Instantiated Netlists
Ref Name	Used
Search
28°C
Smoke
ENG
US
18:26
17-01-2025

Verilog Code:

```
// 32-bit Adder/Subtractor using `+` Operator
module operator_adder_subtractor(
    input [31:0] a,
    input [31:0] b,
    input mode,  // mode = 0 for addition, mode = 1 for subtraction
    output [31:0] sum,
    output carry_out
);
    assign {carry_out, sum} = mode ? a - b : a + b;
endmodule
```

Test Bench Code:

```
// Testbench for 32-bit Adder/Subtractor using `+` Operator
module tb_operator_adder_subtractor;
    reg [31:0] a, b;
    reg mode;
    wire [31:0] sum;
    wire carry_out;

    // Instantiate Module
    operator_adder_subtractor uut (
        .a(a),
        .b(b),
        .mode(mode),
        .sum(sum),
        .carry_out(carry_out)
    );

    initial begin
```

// **Test Cases**

$display("Time\tMode\tA\tB\tSum\tCarry");

$monitor("%0t\t%b\t%h\t%h\t%h\t%b", $time, mode, a, b, sum, carry_out);

a = 32'h00000001; b = 32'h00000001; mode = 0; #10; // **Addition**

a = 32'hFFFFFFFF; b = 32'h00000001; mode = 0; #10; // **Addition with overflow**

a = 32'h00000010; b = 32'h00000001; mode = 1; #10; // **Subtraction**

a = 32'h00000010; b = 32'h00000020; mode = 1; #10; // **Subtraction with negative result**

$finish;

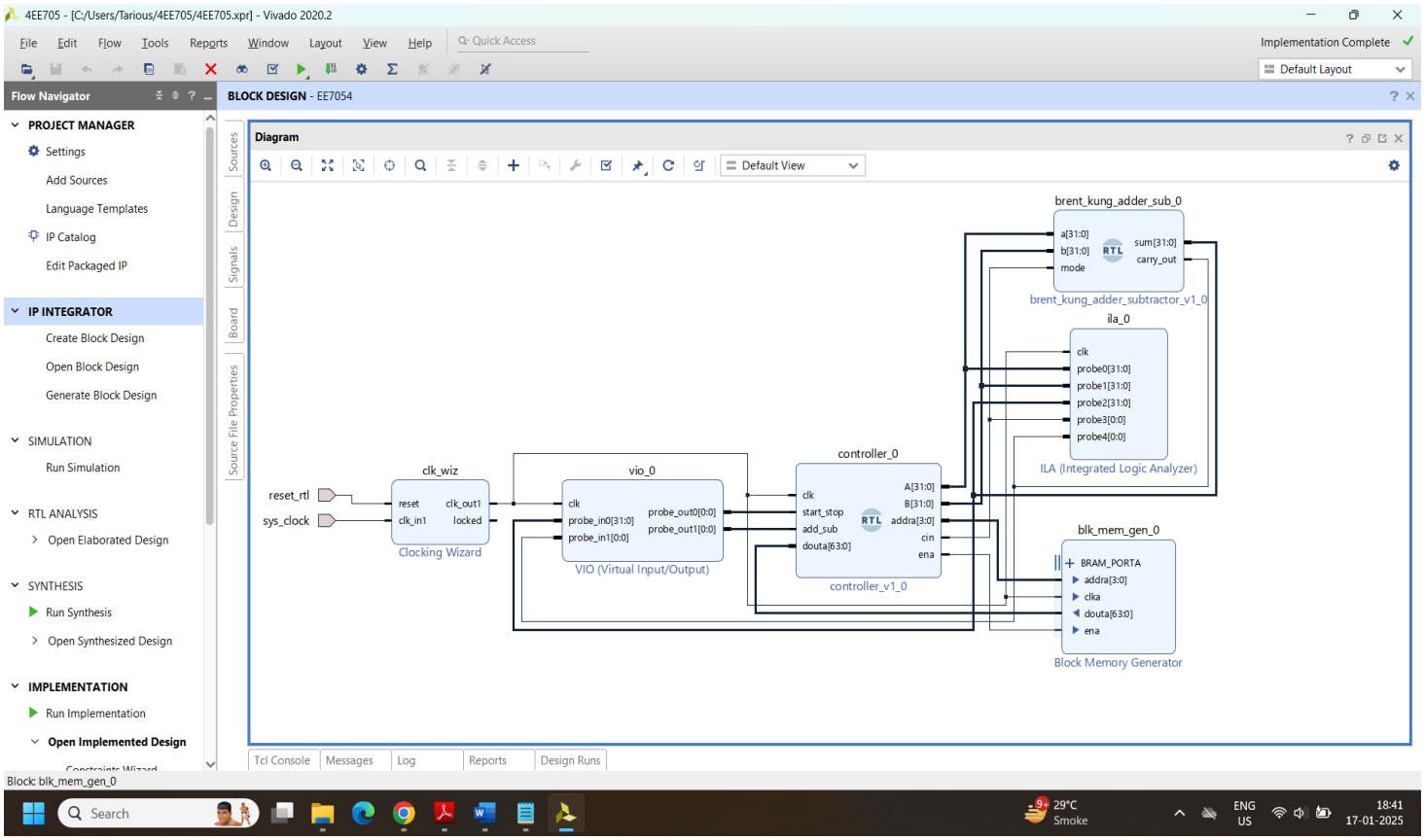
end

endmodule

Task 2: Design a controller which takes 2 signals from VIO: (a) A start/stop signal which can continuously read the data from BRAM and send it to the Adder (b) A signal to select addition or subtraction operation to be performed. BRAM will provide the inputs to your Adder as shown in the figure given below.

Create a testbench to test the controller with BRAM.

Block Design



Constraint File Code:

```
#system clk H16 - 125MHz
set_property -dict {PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports sys_clock]

#Inputs en and reset from slider switches
set_property -dict {PACKAGE_PIN M20 IOSTANDARD LVCMOS33} [get_ports reset_rtl]
```

Verilog Code for Controller:

```
`timescale 1ns / 1ps
module controller (
    input clk,              // Clock signal
    input start_stop,       // Start/Stop signal
    input add_sub,          // Add/Subtract mode (0 = Add, 1 = Subtract)
    input [63:0] douta,     // Data output from BRAM
    output reg [31:0] A,    // Operand A
    output reg [31:0] B,    // Operand B
    output reg [3:0] addra = 4'b0000,  // Address to BRAM
    output reg cin,
    output reg ena  // Enable signal
            // Carry-in to the adder (add_sub passed here)
);
    // Sequential block for managing operations
    always @(posedge clk) begin
                if (start_stop) begin
              ena<=1;
                  A <= douta[63:32];    // Upper 32 bits of douta
              // Set carry-in based on add_sub (0 for Add, 1 for Subtract)
              if (add_sub)
              begin
              B <= ~douta[31:0];
```

```
            cin <= 1;
        end
        else
            begin
            B <= douta[31:0];
            cin <= 0;
            end
         // Increment address
        addra <= addra + 1;
    end
    else begin
        // If start/stop is low, reset A, B, and cin while holding the address
        A <= 32'd0;
        B <= 32'd0;
        cin <= 1'b0;
        ena<=0;
    end
  end
endmodule
```

Test Bench Code for Controller

```
`timescale 1ns / 1ps
module controller_tb;
    // Inputs to the controller
    reg clk;
    reg start_stop;
    reg add_sub;
    reg [63:0] douta;
```

```
// Outputs from the controller
  wire [31:0] A;
  wire [31:0] B;
  wire [3:0] addra;
  wire cin;
  wire ena;

  // Instantiate the controller module
  controller uut (
      .clk(clk),
      .start_stop(start_stop),
      .add_sub(add_sub),
      .douta(douta),
      .A(A),
      .B(B),
      .addra(addra),
      .cin(cin),
      .ena(ena)
  );
  // Generate clock signal
  always begin
      #5 clk = ~clk; // Clock period is 10ns
  end
  // Stimulus
  initial begin
      // Initialize inputs
      clk = 0;
      start_stop = 0;
      add_sub = 0;
      douta = 64'h0000000000000000;
```

```
    // Test case 1: Start the operation with Add mode (add_sub = 0)
    #10;
    douta = 64'h1234567890ABCDEF;  // Test data for douta
    start_stop = 1;
    add_sub = 0;  // Add operation
    #20;

    // Check values of A, B, and other outputs
    $display("A = %h, B = %h, addra = %h, cin = %b, ena = %b", A, B, addra, cin, ena);

    // Test case 2: Start the operation with Subtract mode (add_sub = 1)
    #10;
    douta = 64'hFEDCBA9876543210;  // Test data for douta
    add_sub = 1;  // Subtract operation
    #20;

    // Check values of A, B, and other outputs
    $display("A = %h, B = %h, addra = %h, cin = %b, ena = %b", A, B, addra, cin, ena);

    // Test case 3: Stop the operation
    #10;
    start_stop = 0;  // Stop the operation
    #20;
    // Final check
    $display("A = %h, B = %h, addra = %h, cin = %b, ena = %b", A, B, addra, cin, ena);

    // End the simulation
    $finish;
  end
endmodule
```