Assignment 4 Report

Name: Adarsh Sinha **Stream:** Electronic Systems

Roll No: 21307r005 **Dept.:** Electrical Engineering

Contents of the folder:

1. **assignment-4.pdf**: A copy of Assignment-4 Questionnaire
2. **Proj_Brent32:** The main quartus prime project folder.
3. **Proj_Brent32/Ety_BrentKung32.vhd**: VHDL file for Brent-Kung 32 bit adder.
4. **Proj_Brent32/Ety_Testbench.vhd:** VHDL file for Testbench
5. **Proj_Brent32/TestData.txt:** Contains the data input and reference result for the test bench. It is in a format more convenient for program to read.
6. **Proj_Brent32/TestData_separateVisible.txt:** Contains the data input and results for the test bench. It is in a format more readable for programmer.
7. **Results:** Snapshot of Results obtained
8. **Vlsi Assig4CodeExplaination.mp4**: Video explanation of the code

Task: Make 32-bit Brent-Kung Adder

Theory and reference formulae:

- 1st Order formula for carry: $C_{i+1} = G_i + P_i \cdot C_i$
- 2nd Order formula for carry: $C_{2i+2} = G^2_{2i+1,2i} + P^2_{2i+1,2i} \cdot C_{2i}$
- 3rd Order formula for carry: $C_{4i+4} = G^3_{4i+3,4i} + P^2_{4i+3,4i} \cdot C_{4i}$
- 4th Order formula for carry: $C_{8i+8} = G^4_{8i+7,8i} + P^4_{8i+7,8i} \cdot C_{8i}$
- 5th Order formula for carry: $C_{16i+16} = G^5_{16i+15,16i} + P^5_{16i+15,16i} \cdot C_{16i}$
- 6th Order formula for carry: $C_{32i+32} = G^6_{32i+31,32i} + P^6_{32i+31,32i} \cdot C_{32i}$

Using the Above mentioned reference formulae we can divide the Carries into following groups:

- Each Group is made based on when in the time, the variables needed to calculate these carries are available.
  - Group 1: C_1, C_2, C_4, C_8, C_{16}, C_{32}.
  - Group 2: C_3, C_5, C_6, C_9, C_{10}, C_{12}, C_{17}, C_{18}, C_{20}, C_{24}.
  - Group 3: C_7, C_{11}, C_{13}, C_{14}, C_{19}, C_{21}, C_{22}, C_{25}, C_{26}, C_{28}.
  - Group 4: C_{15}, C_{23}, C_{27}, C_{29}, C_{30}.

- ○ Group 5: C_{31}

RTL View of the adder:

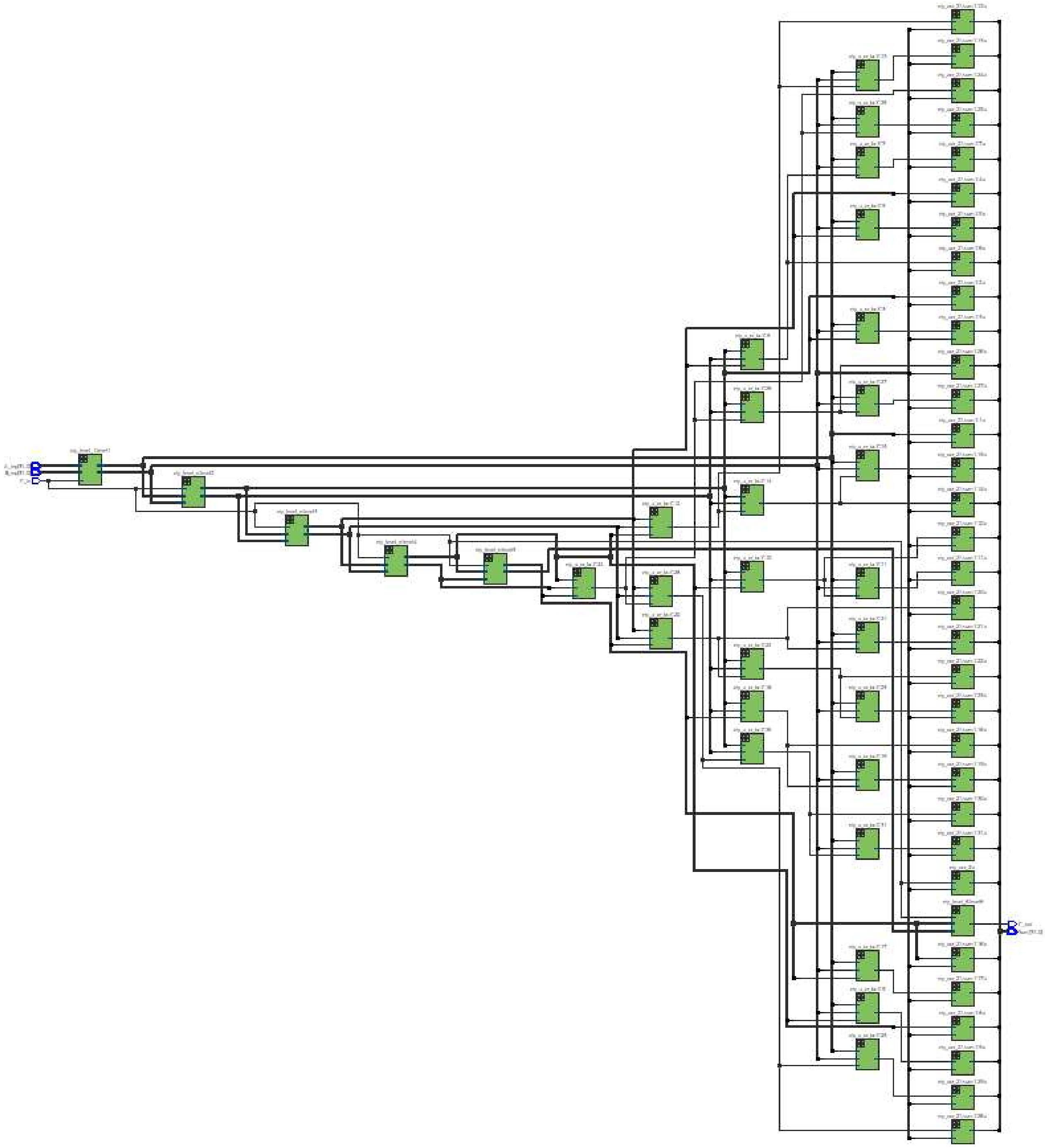


Figure 1: Zoomed out RTL view of the synthesised Brent-Kung 32 bit adder

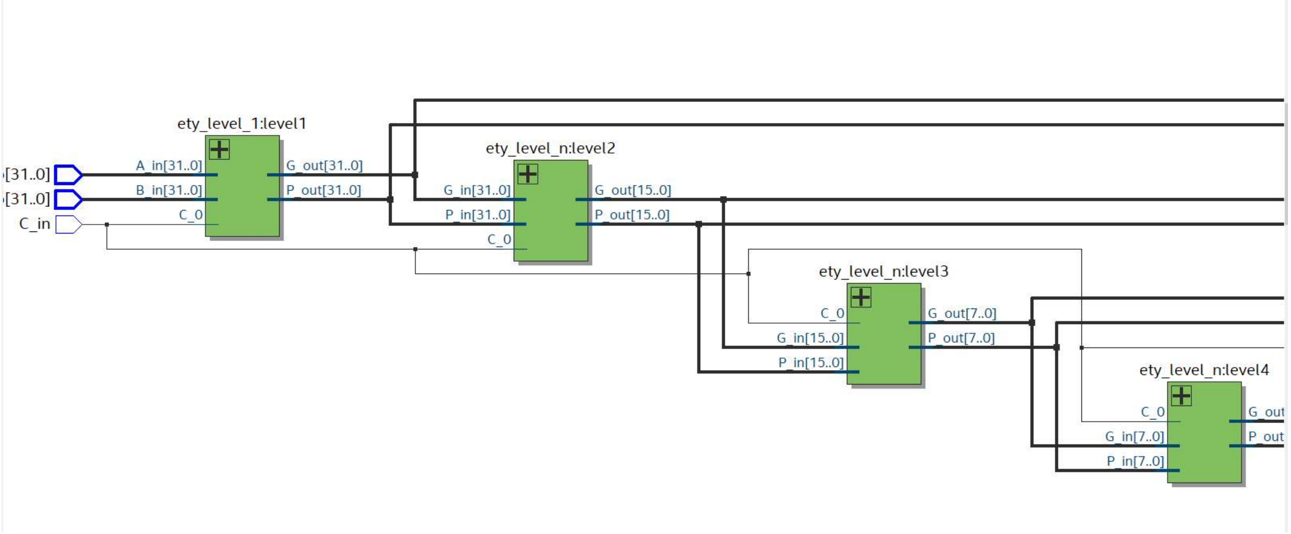
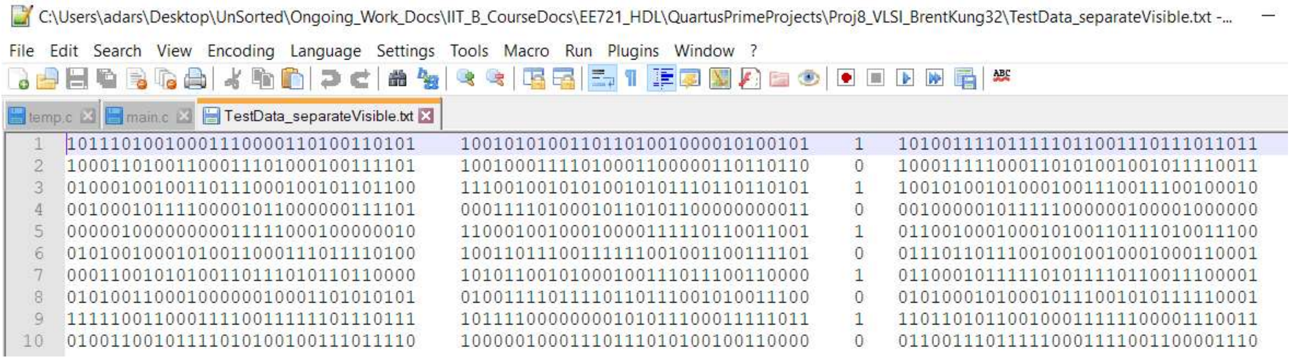


Figure 2: Encapsulated Levels of the Brunt-Kung Adder

Testbench input data arrangement


C:\Users\adars\Desktop\UnSorted\Ongoing_Work_Docs\IIT_B_CourseDocs\EE721_HDL\QuartusPrimeProjects\Proj8_VLSI_BrentKung32\TestData_separateVisible.txt -...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
temp.c
main.c
TestData_separateVisible.txt

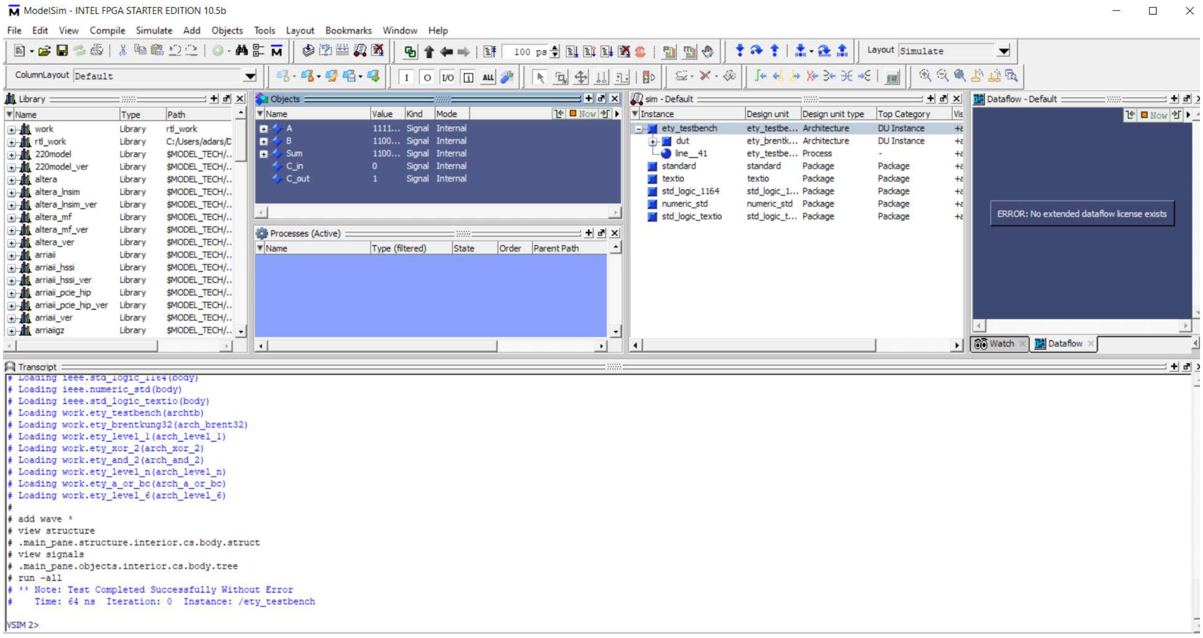
```
10111010010001110000110100110101     10010101001101101001000010100101     1     10100111101111101100111011011011
10001101001100011101000100111101     10010001111010001100000110110110     0     10001111100011010100100101110011
01000100100110111000100101101100     11100100101010010101110110110101     1     10010100101000100111001110010010
00100010111100001011000000111101     00011110100010110101100000000011     0     00100000101111100000010000100000
00000100000000011111000100000010     11000100100010000111110110011001     1     01100100010001010011011101001110
01010010001010011000111011110100     10011011100111111001001100111101     0     01110110111001001001000100011000
00011001010100110111010110110000     10101100101000100111011100110000     1     01100010111110101111011001110000
01010011000100000010001101010101     01001111011110110111001010011100     0     01010001010001011100101011110001
11111001100011110011111101110111     10111100000000101011100011111011     1     11011010110010001111110000111001
01001100101111010100100111011110     10000010001110111010100100110000     0     01100111011111100011110011000011
```

Figure 3: Input and Reference results for Brunt-Kung Adder

- The First Column is the 32 bit input-A for the adder
- The Second Column is the 32 bit input-B for the adder
- The Third Column is the 1 bit input-Carry for the adder
- The Fourth Column is the 33 bit reference output for input-A + input-B + input-Carry
- There are 64 data points randomly generated contained in the file.

Result:

The Adder passes all the test inputs provided indicating proper functioning. Following are the results of the ModelSim Simulation



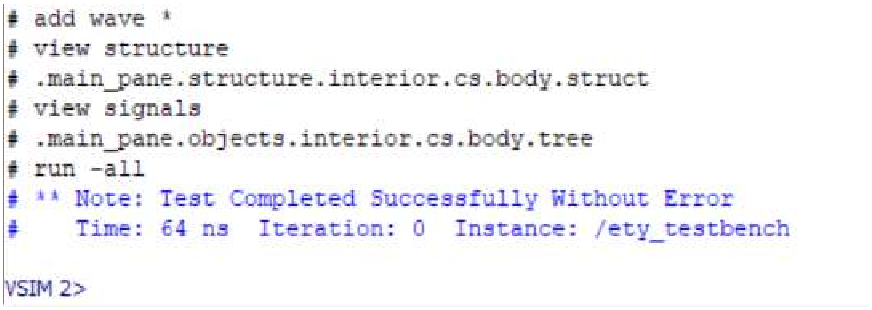
ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Objects Tools Layout Bookmarks Window Help
Library
Objects
Processes (Active)
sim - Default
Dataflow - Default
ERROR: No extended dataflow license exists
Transcript

```
# Loading ieee.std_logic_1164(body)
# Loading ieee.numeric_std(body)
# Loading ieee.std_logic_textio(body)
# Loading work.ety_testbench(archtb)
# Loading work.ety_brentkung32(arch_brent32)
# Loading work.ety_level_1(arch_level_1)
# Loading work.ety_xor_2(arch_xor_2)
# Loading work.ety_and_2(arch_and_2)
# Loading work.ety_level_n(arch_level_n)
# Loading work.ety_a_or_bc(arch_a_or_bc)
# Loading work.ety_level_6(arch_level_6)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: Test Completed Successfully Without Error
#    Time: 64 ns  Iteration: 0  Instance: /ety_testbench

VSIM 2>
```

Figure 4: ModelSim View

Proof of successful completion of testbench:

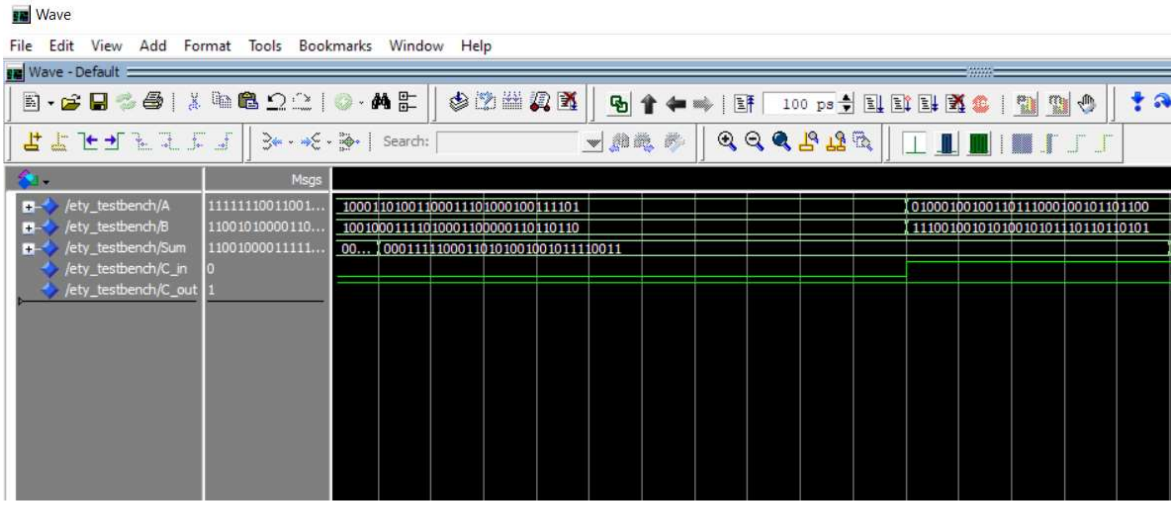


```
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: Test Completed Successfully Without Error
#    Time: 64 ns  Iteration: 0  Instance: /ety_testbench

VSIM 2>
```

Figure 5: ModelSim Test Completion

Wave View of the result:



```
/ety_testbench/A      11111110011001...  10001101001100011101000100111101  01000100100110111000100101101100
/ety_testbench/B      1100101000011O...  10010001111010001100000110110110  11100100101010010101110110110101
/ety_testbench/Sum    11001000011111...  00... 00011111000110101001001011110011
/ety_testbench/C_in   0
/ety_testbench/C_out  1
```

Figure 6: ModelSim Wave View

Observations:

- The Brent-Kung adder took 800 Pico Seconds to calculate One 32 bit sum, where each logic block takes 100 Pico Seconds of Delay. Therefore for 6 stages we should get around 600 picoseconds and some additional delay because the sum is generated after the carry is calculated.
- The Adder trades hardware resources for speed.