**Computer Logic – Practical 2**

**Objective:**

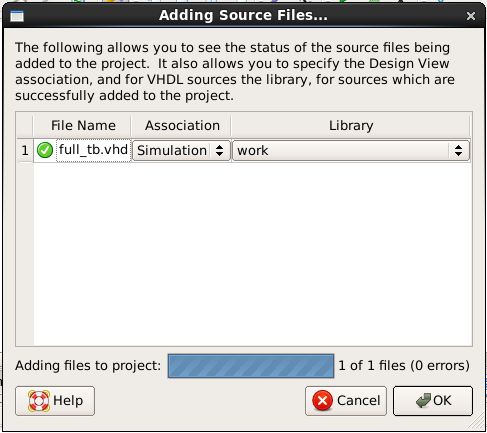
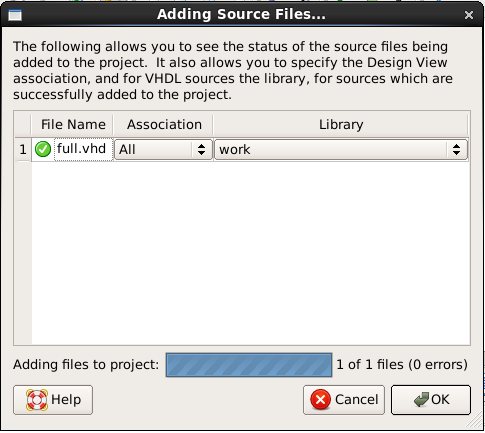
To build, test and compare a carry-ripple adder and a carry-lookahead adder.

**Tasks:**

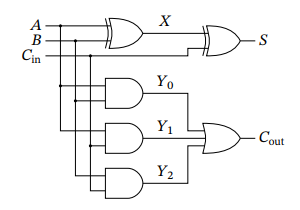
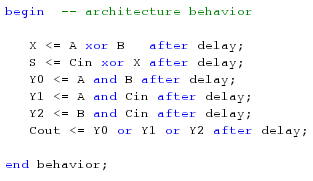
1. A new project called “*Project2*” was created using the same procedure as used in “*Project1*”.
2. The following files where then copied to the project directory:

* *full.vhd*
* *full\_tb.vhd*
* *ripple.vhd*
* *ripple\_tb.vhd*
* *full\_lookahead.vhd*
* *full\_lookahead\_tb.vhd*
* *lookahead.vhd*
* *lookahead\_tb.vhd*

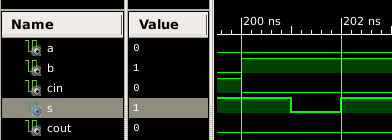
1. The 8 files above were then added to the project with the menu item *Project: Add Source…* The association was set to *Simulation* for files with “\_tb” in their filename. Those without “\_tb” in the filename were associated to *All* as shown:

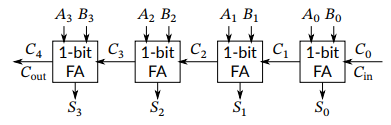
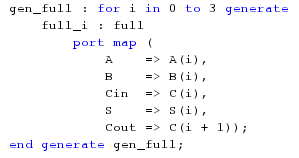
1. The file *full.vhd* defines the single-bit full adder shown below. Statements for X and S were already written in this file and both contained a delay clause. Statements for Y0 , Y1 , Y2 and Cout were then added to complete the full-adder. Similar delay clauses to those of X and S were implemented in the VHDL code.

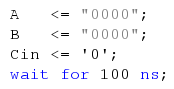
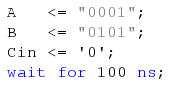
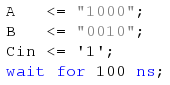
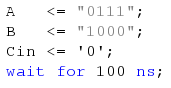
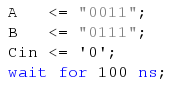
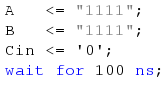
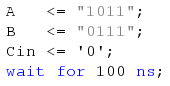
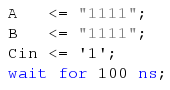
1. The *full\_tb.vhd* file defining a testbench for the full adder was simulated using behavioral simulation and the output result was analysed. The emulated gate delay was known to be 1 ns for each gate whilst the propagation delays for the outputs S and Cout were found to be 2 ns as shown in one instance between 200 and 202 ns.



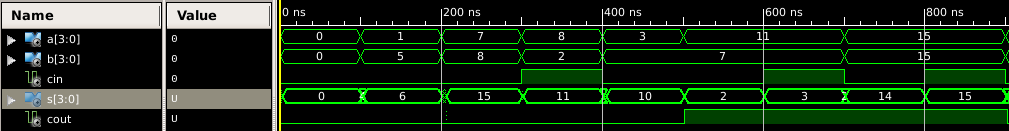
1. In this step, the entity *ripple.vhd* defining the four-bit ripple-carry adder in the diagram was looked into. This particular task required observing the VHDL code only, especially the part where the four full adders were instantiated.

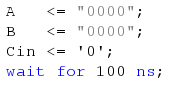
1. The test bench *ripple\_tb.vhd* was used to simulate inputs for the testing of the carry-ripple adder. This time, these inputs were not tested exhaustively. The following additions were performed to try out the test bench with the first addition having all inputs as 0s and the final one having all inputs as 1s:

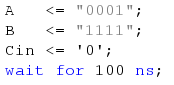
All these input statements were chosen carefully to try out as many different cases as possible. The sum and carry outputs were then checked to see if they were as expected.



1. In this step, the propagation delay of Cout for the ripple-carry adder was measured as follows:
2. An addition operation with operands A = 00002 , B = 00002 , and Cin = 0 was needed to be added so that C0, C1, C2, C3 and C4 all equal 0. However this was already part of the code as it was one of the operands being tested in step 7.

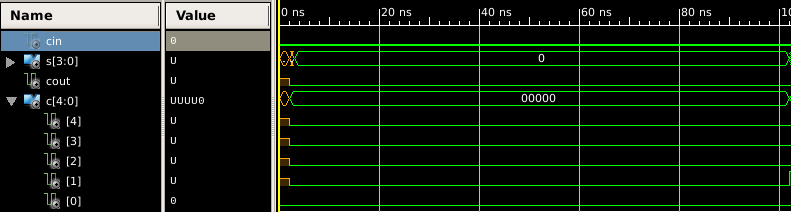


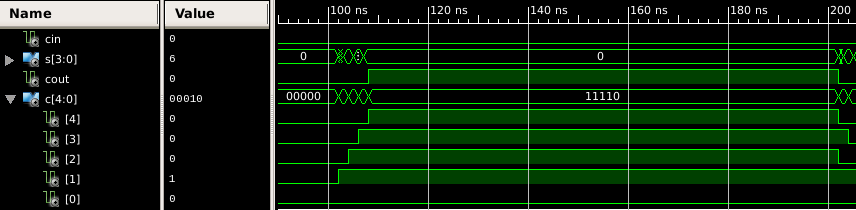
1. Another addition operation with operands A = 00012 , B= 11112 , and Cin = 0 was needed to be added so that the carry propagates through all the internal carries. This operand was not part of the previous operands being tested in step 7 so it was added at this stage.



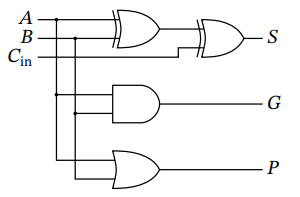
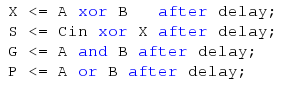
1. The test bench was simulated once again to observe the internal propagation delay of the ripple. This was done by dragging the object *C [4:0 ]* from the *instances pane: ripple\_tb: uut* to the waveform pane. The added waveforms were then made visible once the simulation was reloaded again.

The screenshots below show the different cases. The first one represents case (a) in which all inputs were 0 and there was no need for the carry to propagate at all. The second screenshot represents case (b) in which the carry was required to propagate through all internal carries. This was done with a propagation delay of 2 ns per propagation.

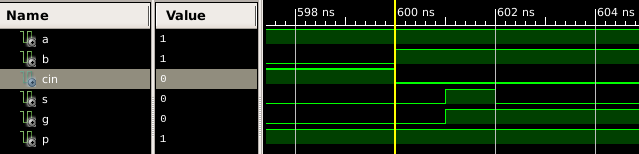


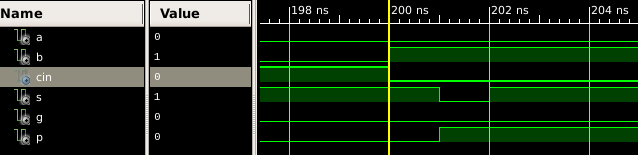


1. The file *full\_lookahead.vhd* contains a single-bit full-adder for a carry-lookahead adder shown. Statements for G and P were then added to complete the single-bit full-adder.

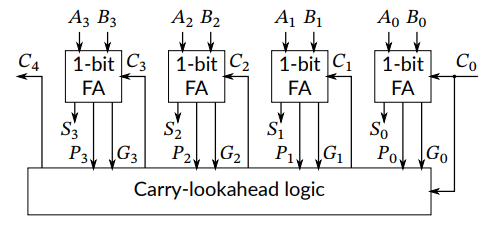
 

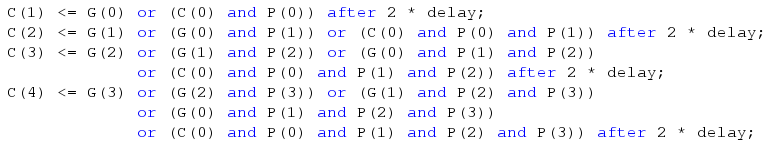
1. The file *full\_lookahead\_tb.vhd* contains a test bench for the single-bit carry-lookahead adder. This file was simulated and the propagation delays for G and P were measured. These were both found to be 1 ns. The screenshots below show proof for this with the first one showing G and the second, P.



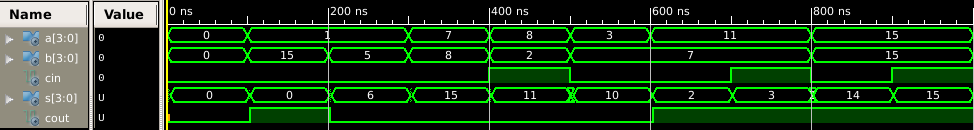


1. The file *lookahead.vhd* containing an implementation of the carry-lookahead adder shown was modified by including statements for C2, C3 and C4.

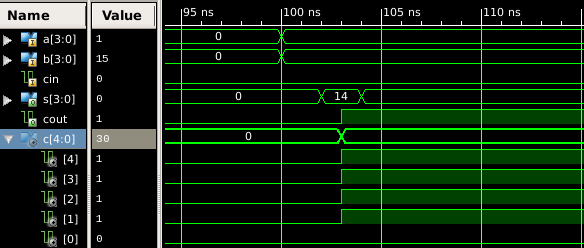




1. The implementation of the test bench *lookahead\_tb.vhd* was completed by copying the complete stimulus process from *ripple\_tb.vhd*. Then, each output sum was checked for correctness sake.

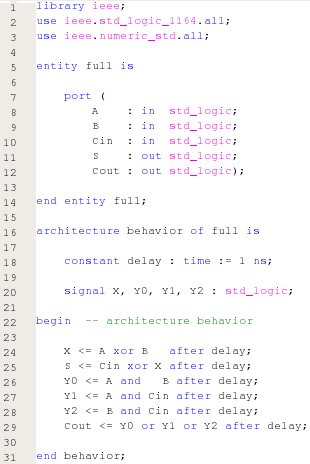


1. The maximum propagation carry delay for the lookahead adder was found to be 3 ns as shown in the screenshot. This is because the signal would have to pass through 3 gates having a propagation delay of 1 ns delay per gate. Comparing this to step 8, we see that all the Couts in this case were computed at the same time since one C­out didn’t lead on to the other. On the other hand, in step 8 the propagation delay was of 2 ns per Cout since one input would lead onto the other.

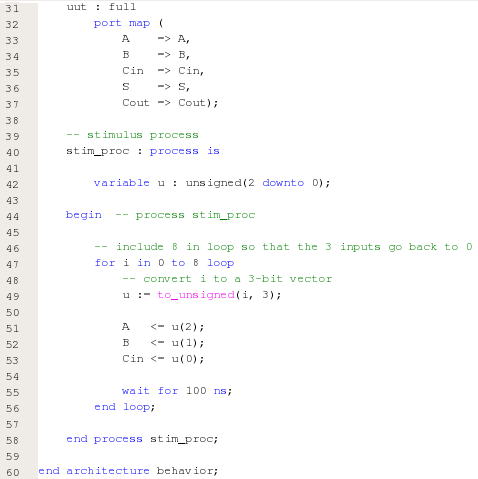
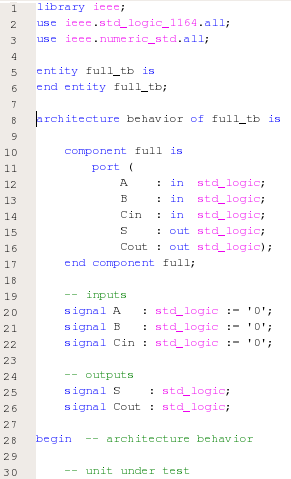


**Appendix (Code):**

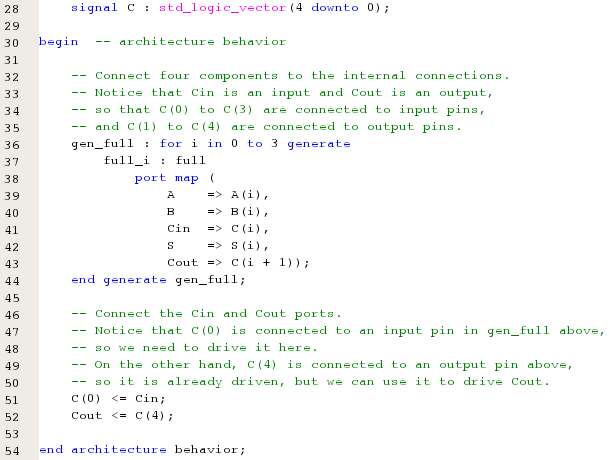
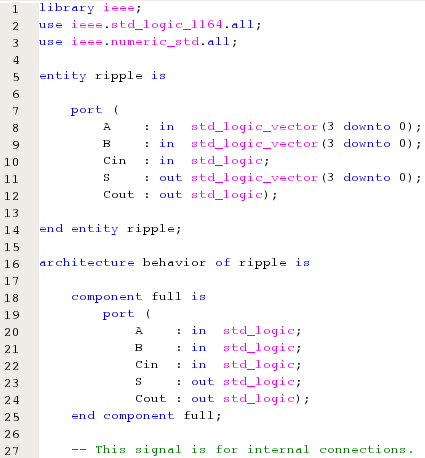
***full.vhd:***



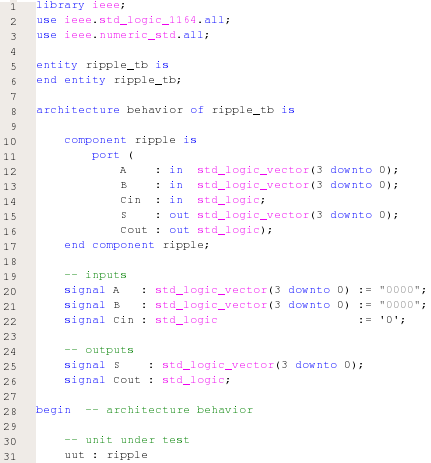
***full\_tb.vhd:***

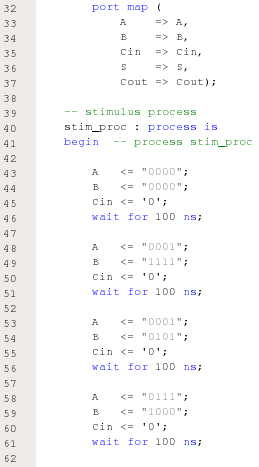


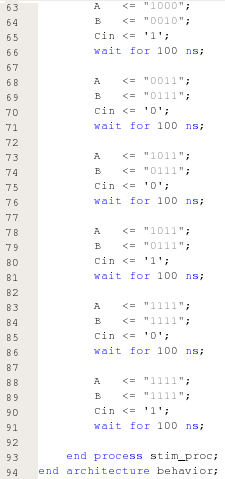
***ripple.vhd***



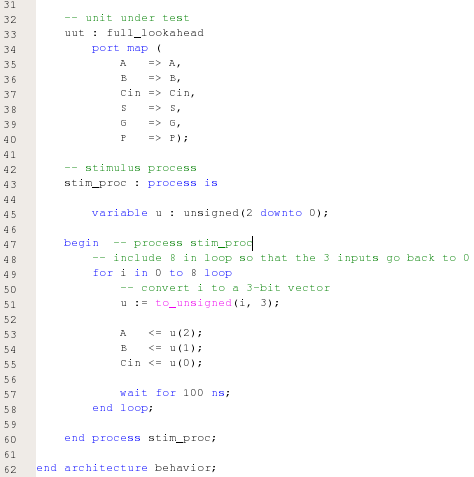
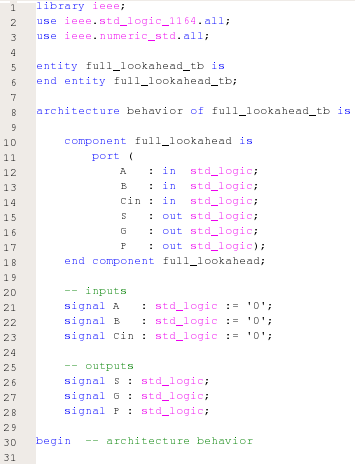
***ripple\_tb.vhd****:*



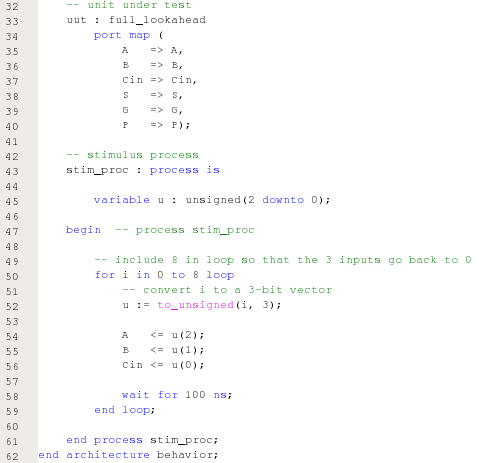
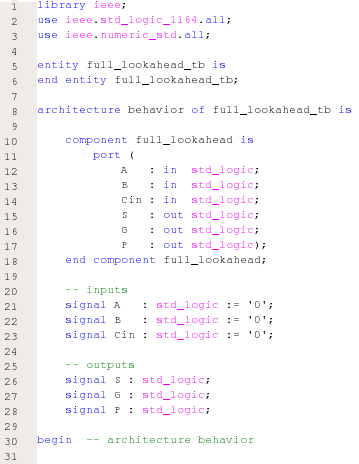




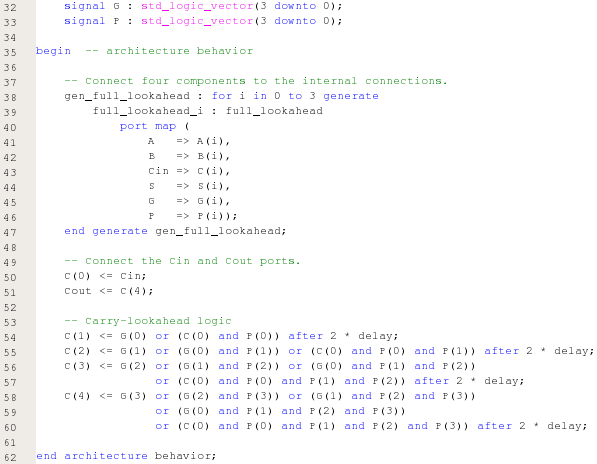
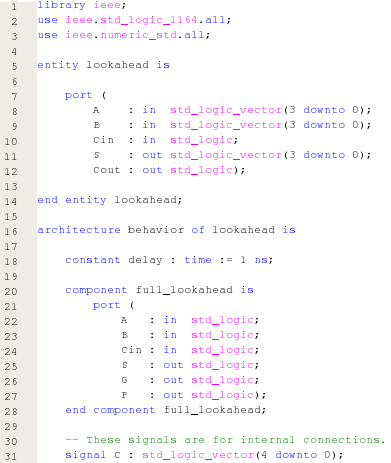
***full\_lookahead.vhd:***



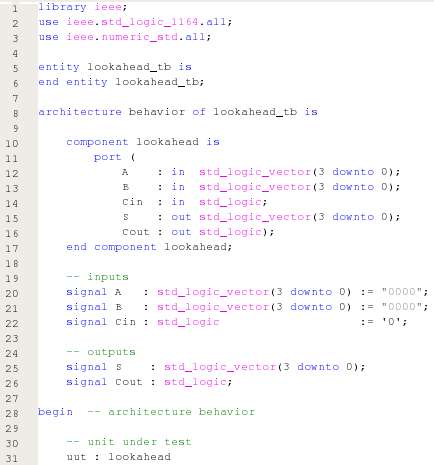
***full\_lookahead\_tb.vhd:***

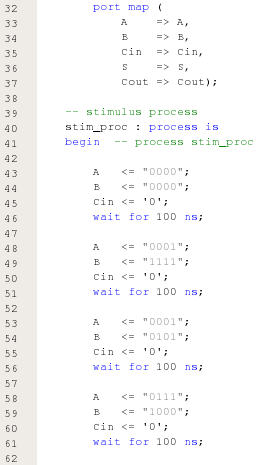


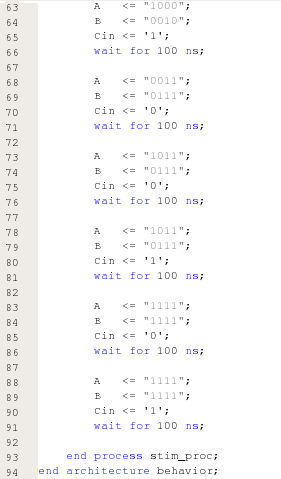
***lookahead.vhd:***



***lookahead\_tb.vhd:***

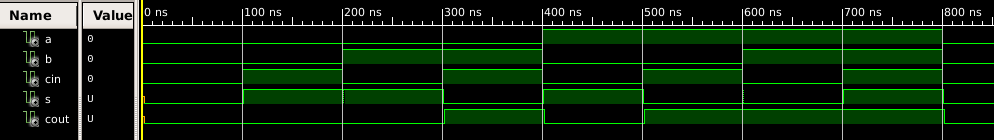




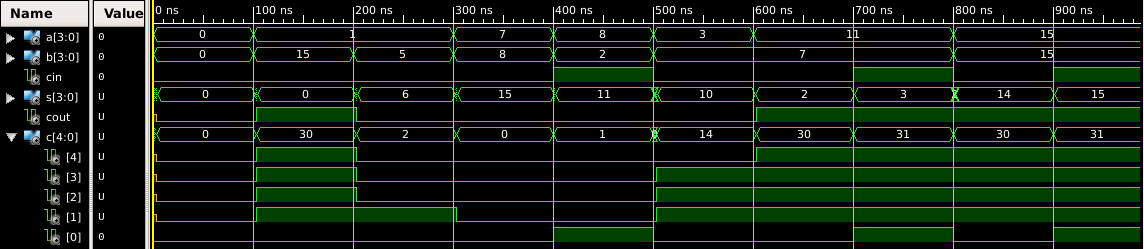


***Simulated Behavioral Models:***

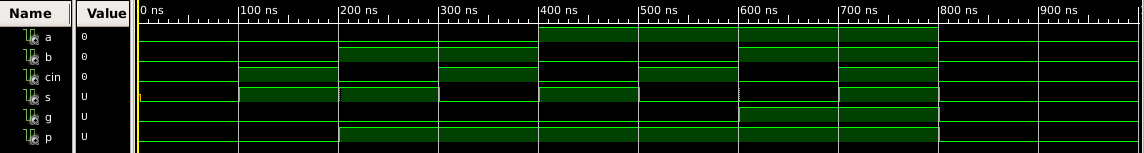
***full\_tb.vhd:***



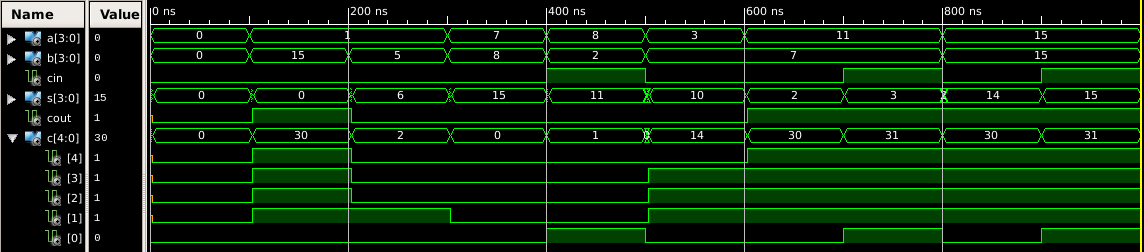
***ripple\_tb.vhd:***



***full\_lookahead\_tb.vhd:***



***lookahead\_tb.vhd:***



***Conclusion:***

A carry-ripple adder and a carry-lookahead adder were successfully built, tested and compared.