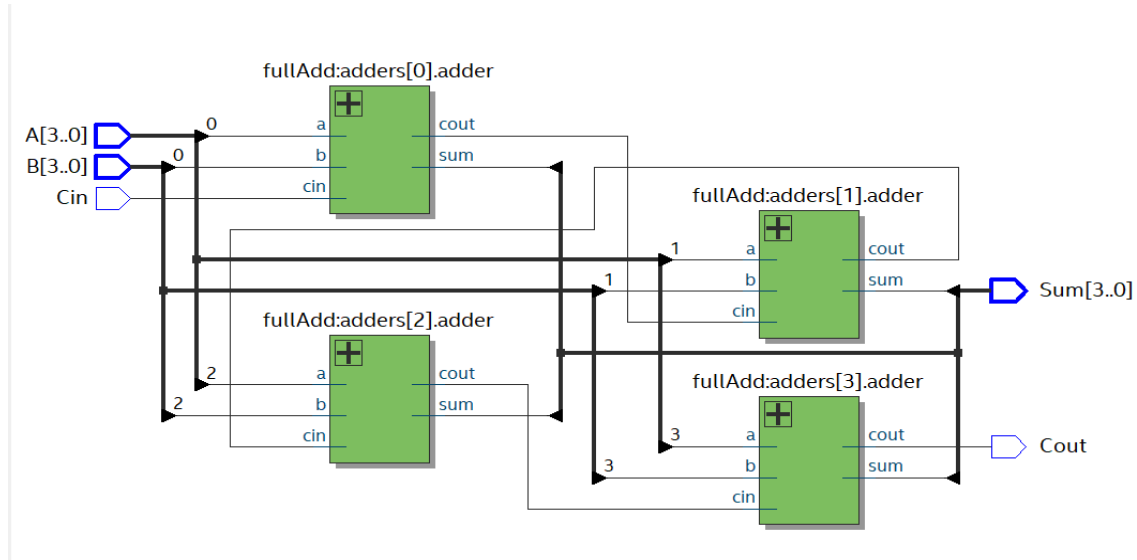


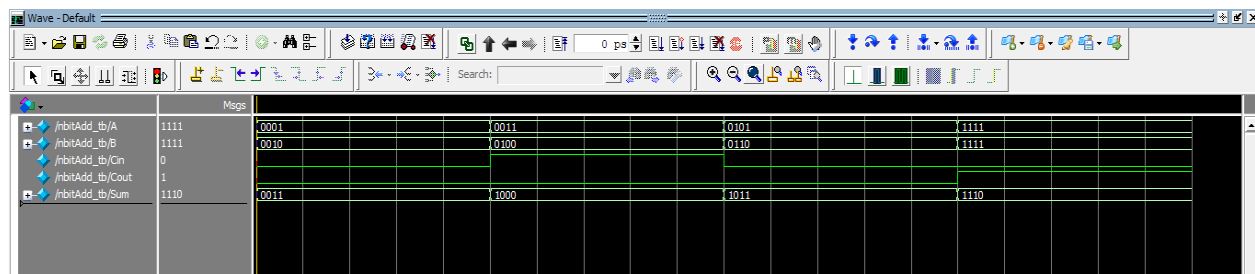
## 4. An N-Bit Full Adder

### Design:



- ❖ Designed a single sub-module named “fullAdd” having single bit ports ‘a’, ‘b’, ‘cin’ as input and ‘sum’ and ‘cout’ as outputs.
- ❖ Designed a top level module named “nbitAdd” having N bits ports ‘A’, ‘B’, and ‘Cin’ as inputs and ‘Sum’ and ‘Cout’ as outputs.
- ❖ The ‘nbitAdd’ top module is using “generate” block and an instance of “fullAdd” module to generate N hardware replicas for Nbit addition.

### Verification:



```
# Applying four different sets of input values
# A = 0001, B = 0010, Cin = 0, Sum = 0011, Cout = 0, expected Sum = 0011, expected Cout = 0
# PASS
# A = 0011, B = 0100, Cin = 1, Sum = 1000, Cout = 0, expected Sum = 1000, expected Cout = 0
# PASS
# A = 0101, B = 0110, Cin = 0, Sum = 1011, Cout = 0, expected Sum = 1011, expected Cout = 0
# PASS
# A = 1111, B = 1111, Cin = 0, Sum = 1110, Cout = 1, expected Sum = 1110, expected Cout = 1
# PASS
```

- ❖ Designed the testbench that validates a 4-bit ripple-carry adder by applying four distinct input combinations of a, b, and cin.
- ❖ It calculates the expected sum and carry using built-in addition operation (+) and compares them with the module output.
- ❖ The check\_result function block prints pass/fail status for each case, ensuring correct arithmetic behavior.