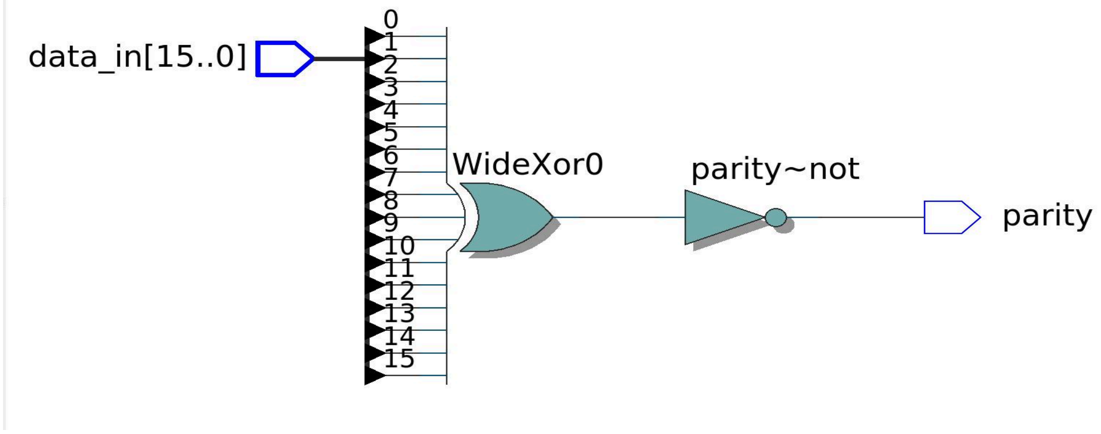
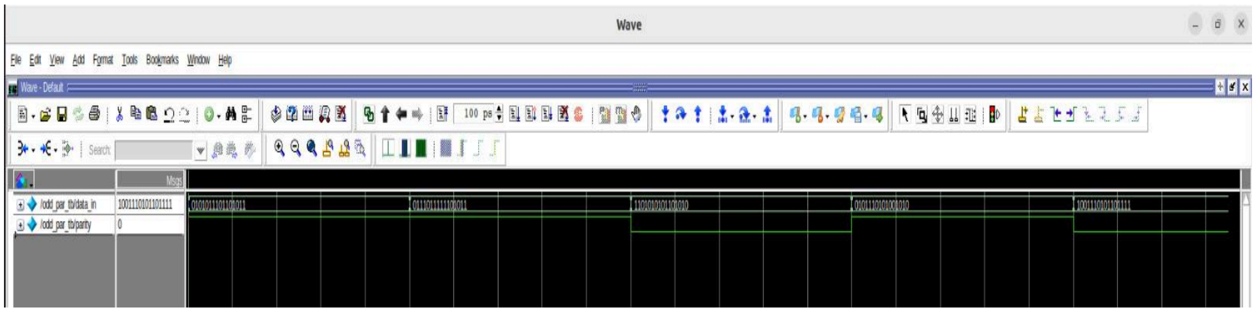
1.Odd parity of a 16-bit number.

- ***Design:***



- ❖ Designed a module named "odd_parity", having ports, 'data_in' as input and 'parity' as output.
- ❖ Applied bitwise XOR reduction operator to the input 16bit data to achieve a resultant single bit intermediate output.
- ❖ Inverted the resultant bit to achieve an "Odd Parity Bit" as output.

- ***Verification:***



```
# Sending a set of 5 data values to the design and checking if testcase passes or fails
# data = 0101011101101011, parity = 1, expected = 1
# PASS
# data = 0111011111101011, parity = 1, expected = 1
# PASS
# data = 1101010101101010, parity = 0, expected = 0
# PASS
# data = 0101110101001010, parity = 1, expected = 1
# PASS
# data = 1001110101101111, parity = 0, expected = 0
# PASS
```

- This testbench verifies the “`odd_parity`” module by applying five 16-bit binary input patterns.
- It calculates the expected odd parity using bitwise XOR inside a function block and compares it with the module’s output.
- Each result is printed with a pass/fail message to confirm correct parity generation logic.
- This validates the module’s ability to detect or generate odd parity across varied input cases.

  .