Testbenching Report for ripple_carry_adder

Table of Contents

Testbench Summary

Component	Total Tests	Passed	Failed
ripple_carry_adder_N1	8	8	0
ripple_carry_adder_N2	32	32	0
ripple_carry_adder_N3	35	35	0
ripple_carry_adder_N4	35	35	0
ripple_carry_adder_N5	35	35	0
ripple_carry_adder_N6	35	35	0
ripple_carry_adder_N7	35	35	0
ripple_carry_adder_N8	35	35	0
full_adder_	8	8	0
half_adder_	4	4	0

Testbench for ripple_carry_adder with parameter(s) N1

Total tests: 8

Passed tests: 8

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
3	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
4	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
5	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
6	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
0	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
7	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
2	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
            def matches(self, filename):
                return self.pattern in filename
```

Generate expected values function:

```
            def generate_expected(self, test_case):
                max_val = (1 << self.bit_width) - 1
                if "cin" in test_case:
                    sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
                    outs = {
                        "sum": sum_val & max_val,
                        "cout": sum_val >> self.bit_width
                    }
                else:
                    sum_val = test_case["a"] + test_case["b"]
                    outs = {
                        "sum": sum_val & max_val,
                        "cout": sum_val >> self.bit_width
                    }
                return outs
```

Testbench for ripple_carry_adder with parameter(s) N2

Total tests: 32

Passed tests: 32

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
27	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
11	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
9	00 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
29	00 (bin) / 0 (dec)	10 (bin) / 2 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
8	00 (bin) / 0 (dec)	01 (bin) / 1 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
16	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
10	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
15	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
5	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
24	00 (bin) / 0 (dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
21	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
22	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
13	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
14	00 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
12	00 (bin) / 0 (dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
20	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
31	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
7	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
4	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
18	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
6	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
17	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
2	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
28	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
        def matches(self, filename):
            return self.pattern in filename
```

Generate expected values function:

```
        def generate_expected(self, test_case):
            max_val = (1 << self.bit_width) - 1
            if "cin" in test_case:
                sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            else:
                sum_val = test_case["a"] + test_case["b"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            return outs
```

Testbench for ripple_carry_adder with parameter(s) N3

Total tests: 35

Passed tests: 35

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
14	010 (bin) / 2 (dec)	001 (bin) / 1 (dec)	1 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
19	110 (bin) / 6 (dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
30	000 (bin) / 0 (dec)	010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
10	100 (bin) / 4 (dec)	110 (bin) / 6 (dec)	0 (bin) / 0 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
31	101 (bin) / 5 (dec)	101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
16	111 (bin) / 7 (dec)	110 (bin) / 6 (dec)	0 (bin) / 0 (dec)	101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
7	010 (bin) / 2 (dec)	010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
21	100 (bin) / 4 (dec)	011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
27	000 (bin) / 0 (dec)	000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	001 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
34	001 (bin) / 1 (dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
18	011 (bin) / 3 (dec)	000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
25	110 (bin) / 6 (dec)	011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
20	010 (bin) / 2 (dec)	100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
22	100 (bin) / 4 (dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
17	001 (bin) / 1 (dec)	011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
15	001 (bin) / 1 (dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	011 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
23	110 (bin) / 6 (dec)	111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
11	011 (bin) / 3 (dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
12	110 (bin) / 6 (dec)	001 (bin) / 1 (dec)	1 (bin) / 1 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
5	010 (bin) / 2 (dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
24	011 (bin) / 3 (dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
33	000 (bin) / 0 (dec)	011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	011 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
28	101 (bin) / 5 (dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
26	110 (bin) / 6 (dec)	111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
8	011 (bin) / 3 (dec)	000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	011 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
        def matches(self, filename):
            return self.pattern in filename
```

Generate expected values function:

```
        def generate_expected(self, test_case):
            max_val = (1 << self.bit_width) - 1
            if "cin" in test_case:
                sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            else:
                sum_val = test_case["a"] + test_case["b"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            return outs
```

Testbench for ripple_carry_adder with parameter(s) N4

Total tests: 35

Passed tests: 35

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
26	1101 (bin) / 13 (dec)	1100 (bin) / 12 (dec)	0 (bin) / 0 (dec)	1001 (bin) / 9 (dec)	9 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
16	0001 (bin) / 1 (dec)	1101 (bin) / 13 (dec)	1 (bin) / 1 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
0	1111 (bin) / 15 (dec)	0010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
19	1110 (bin) / 14 (dec)	1010 (bin) / 10 (dec)	1 (bin) / 1 (dec)	1001 (bin) / 9 (dec)	9 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
23	0011 (bin) / 3 (dec)	0001 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
30	0100 (bin) / 4 (dec)	1110 (bin) / 14 (dec)	0 (bin) / 0 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
20	0000 (bin) / 0 (dec)	1100 (bin) / 12 (dec)	0 (bin) / 0 (dec)	1100 (bin) / 12 (dec)	12 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
33	0000 (bin) / 0 (dec)	1000 (bin) / 8 (dec)	1 (bin) / 1 (dec)	1001 (bin) / 9 (dec)	9 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
27	0111 (bin) / 7 (dec)	1011 (bin) / 11 (dec)	0 (bin) / 0 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
15	0010 (bin) / 2 (dec)	1111 (bin) / 15 (dec)	1 (bin) / 1 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
9	0011 (bin) / 3 (dec)	1010 (bin) / 10 (dec)	0 (bin) / 0 (dec)	1101 (bin) / 13 (dec)	13 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
32	0111 (bin) / 7 (dec)	1000 (bin) / 8 (dec)	1 (bin) / 1 (dec)	0000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
29	1011 (bin) / 11 (dec)	0110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
2	0000 (bin) / 0 (dec)	1110 (bin) / 14 (dec)	0 (bin) / 0 (dec)	1110 (bin) / 14 (dec)	14 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
24	1110 (bin) / 14 (dec)	1101 (bin) / 13 (dec)	1 (bin) / 1 (dec)	1100 (bin) / 12 (dec)	12 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
34	1000 (bin) / 8 (dec)	1011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
12	1101 (bin) / 13 (dec)	0110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
13	0111 (bin) / 7 (dec)	0100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	1100 (bin) / 12 (dec)	12 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
21	0100 (bin) / 4 (dec)	1101 (bin) / 13 (dec)	1 (bin) / 1 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
10	1101 (bin) / 13 (dec)	0101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1	0010 (bin) / 2 (dec)	0100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	0110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
31	0111 (bin) / 7 (dec)	0100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
3	1000 (bin) / 8 (dec)	1001 (bin) / 9 (dec)	1 (bin) / 1 (dec)	0010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
14	0101 (bin) / 5 (dec)	0011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	1000 (bin) / 8 (dec)	8 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
5	1001 (bin) / 9 (dec)	0111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	0001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
def matches(self, filename):
    return self.pattern in filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << self.bit_width) - 1
    if "cin" in test_case:
        sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
        outs = {
            "sum": sum_val & max_val,
            "cout": sum_val >> self.bit_width
        }
else:
    sum_val = test_case["a"] + test_case["b"]
    outs = {
        "sum": sum_val & max_val,
        "cout": sum_val >> self.bit_width
    }
return outs
```

Testbench for ripple_carry_adder with parameter(s) N5

Total tests: 35

Passed tests: 35

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
22	00111 (bin) / 7 (dec)	01001 (bin) / 9 (dec)	1 (bin) / 1 (dec)	10001 (bin) / 17 (dec)	17 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
5	00001 (bin) / 1 (dec)	11101 (bin) / 29 (dec)	1 (bin) / 1 (dec)	11111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
29	00101 (bin) / 5 (dec)	10100 (bin) / 20 (dec)	1 (bin) / 1 (dec)	11010 (bin) / 26 (dec)	26 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
14	11000 (bin) / 24 (dec)	00000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	11001 (bin) / 25 (dec)	25 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
18	00101 (bin) / 5 (dec)	10000 (bin) / 16 (dec)	0 (bin) / 0 (dec)	10101 (bin) / 21 (dec)	21 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
19	01011 (bin) / 11 (dec)	00000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	01011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
16	00000 (bin) / 0 (dec)	01111 (bin) / 15 (dec)	0 (bin) / 0 (dec)	01111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
4	00111 (bin) / 7 (dec)	00001 (bin) / 1 (dec)	0 (bin) / 0 (dec)	01000 (bin) / 8 (dec)	8 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
25	11110 (bin) / 30 (dec)	00100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	00011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
34	11001 (bin) / 25 (dec)	11001 (bin) / 25 (dec)	1 (bin) / 1 (dec)	10011 (bin) / 19 (dec)	19 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
28	11101 (bin) / 29 (dec)	10010 (bin) / 18 (dec)	1 (bin) / 1 (dec)	10000 (bin) / 16 (dec)	16 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
12	10010 (bin) / 18 (dec)	10010 (bin) / 18 (dec)	0 (bin) / 0 (dec)	00100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
7	10001 (bin) / 17 (dec)	00110 (bin) / 6 (dec)	0 (bin) / 0 (dec)	10111 (bin) / 23 (dec)	23 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
6	11110 (bin) / 30 (dec)	11010 (bin) / 26 (dec)	0 (bin) / 0 (dec)	11000 (bin) / 24 (dec)	24 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	00100 (bin) / 4 (dec)	11011 (bin) / 27 (dec)	0 (bin) / 0 (dec)	11111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
15	01101 (bin) / 13 (dec)	11011 (bin) / 27 (dec)	0 (bin) / 0 (dec)	01000 (bin) / 8 (dec)	8 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
10	11110 (bin) / 30 (dec)	10111 (bin) / 23 (dec)	1 (bin) / 1 (dec)	10110 (bin) / 22 (dec)	22 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
20	11000 (bin) / 24 (dec)	11110 (bin) / 30 (dec)	0 (bin) / 0 (dec)	10110 (bin) / 22 (dec)	22 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
31	01111 (bin) / 15 (dec)	10111 (bin) / 23 (dec)	1 (bin) / 1 (dec)	00111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
33	01110 (bin) / 14 (dec)	00010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	10001 (bin) / 17 (dec)	17 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
17	10010 (bin) / 18 (dec)	10100 (bin) / 20 (dec)	0 (bin) / 0 (dec)	00110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
23	10001 (bin) / 17 (dec)	01000 (bin) / 8 (dec)	1 (bin) / 1 (dec)	11010 (bin) / 26 (dec)	26 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
2	01001 (bin) / 9 (dec)	00011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	01101 (bin) / 13 (dec)	13 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
27	01100 (bin) / 12 (dec)	10000 (bin) / 16 (dec)	0 (bin) / 0 (dec)	11100 (bin) / 28 (dec)	28 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
8	10010 (bin) / 18 (dec)	10000 (bin) / 16 (dec)	1 (bin) / 1 (dec)	00011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
        def matches(self, filename):
            return self.pattern in filename
```

Generate expected values function:

```
        def generate_expected(self, test_case):
            max_val = (1 << self.bit_width) - 1
            if "cin" in test_case:
                sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            else:
                sum_val = test_case["a"] + test_case["b"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            return outs
```

Testbench for ripple_carry_adder with parameter(s) N6

Total tests: 35

Passed tests: 35

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
29	010101 (bin) / 21 (dec)	101010 (bin) / 42 (dec)	0 (bin) / 0 (dec)	111111 (bin) / 63 (dec)	63 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
8	011001 (bin) / 25 (dec)	110111 (bin) / 55 (dec)	0 (bin) / 0 (dec)	010000 (bin) / 16 (dec)	16 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
11	100110 (bin) / 38 (dec)	001000 (bin) / 8 (dec)	0 (bin) / 0 (dec)	101110 (bin) / 46 (dec)	46 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
22	101101 (bin) / 45 (dec)	101011 (bin) / 43 (dec)	0 (bin) / 0 (dec)	011000 (bin) / 24 (dec)	24 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1	111001 (bin) / 57 (dec)	101011 (bin) / 43 (dec)	0 (bin) / 0 (dec)	100100 (bin) / 36 (dec)	36 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
14	011001 (bin) / 25 (dec)	110001 (bin) / 49 (dec)	1 (bin) / 1 (dec)	001011 (bin) / 11 (dec)	11 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
31	001000 (bin) / 8 (dec)	001100 (bin) / 12 (dec)	0 (bin) / 0 (dec)	010100 (bin) / 20 (dec)	20 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
34	110100 (bin) / 52 (dec)	110010 (bin) / 50 (dec)	1 (bin) / 1 (dec)	100111 (bin) / 39 (dec)	39 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
13	001110 (bin) / 14 (dec)	001000 (bin) / 8 (dec)	1 (bin) / 1 (dec)	010111 (bin) / 23 (dec)	23 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
30	011101 (bin) / 29 (dec)	000001 (bin) / 1 (dec)	1 (bin) / 1 (dec)	011111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
20	111100 (bin) / 60 (dec)	111011 (bin) / 59 (dec)	0 (bin) / 0 (dec)	110111 (bin) / 55 (dec)	55 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
23	010000 (bin) / 16 (dec)	100110 (bin) / 38 (dec)	0 (bin) / 0 (dec)	110110 (bin) / 54 (dec)	54 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
26	110110 (bin) / 54 (dec)	010101 (bin) / 21 (dec)	0 (bin) / 0 (dec)	001011 (bin) / 11 (dec)	11 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
17	011110 (bin) / 30 (dec)	011010 (bin) / 26 (dec)	1 (bin) / 1 (dec)	111001 (bin) / 57 (dec)	57 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
15	001111 (bin) / 15 (dec)	111111 (bin) / 63 (dec)	1 (bin) / 1 (dec)	001111 (bin) / 15 (dec)	15 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
25	110010 (bin) / 50 (dec)	000100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	110110 (bin) / 54 (dec)	54 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
7	100101 (bin) / 37 (dec)	000111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	101101 (bin) / 45 (dec)	45 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
0	011111 (bin) / 31 (dec)	100110 (bin) / 38 (dec)	1 (bin) / 1 (dec)	000110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
32	100100 (bin) / 36 (dec)	110000 (bin) / 48 (dec)	0 (bin) / 0 (dec)	010100 (bin) / 20 (dec)	20 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
19	111010 (bin) / 58 (dec)	001010 (bin) / 10 (dec)	1 (bin) / 1 (dec)	000101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
10	011000 (bin) / 24 (dec)	111010 (bin) / 58 (dec)	1 (bin) / 1 (dec)	010011 (bin) / 19 (dec)	19 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
4	001100 (bin) / 12 (dec)	110001 (bin) / 49 (dec)	1 (bin) / 1 (dec)	111110 (bin) / 62 (dec)	62 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
2	110010 (bin) / 50 (dec)	110011 (bin) / 51 (dec)	1 (bin) / 1 (dec)	100110 (bin) / 38 (dec)	38 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
16	001001 (bin) / 9 (dec)	100100 (bin) / 36 (dec)	0 (bin) / 0 (dec)	101101 (bin) / 45 (dec)	45 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
3	101110 (bin) / 46 (dec)	001011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	111010 (bin) / 58 (dec)	58 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
        def matches(self, filename):
            return self.pattern in filename
```

Generate expected values function:

```
        def generate_expected(self, test_case):
            max_val = (1 << self.bit_width) - 1
            if "cin" in test_case:
                sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            else:
                sum_val = test_case["a"] + test_case["b"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            return outs
```

Testbench for ripple_carry_adder with parameter(s) N7

Total tests: 35

Passed tests: 35

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
32	0100111 (bin) / 39 (dec)	1001000 (bin) / 72 (dec)	0 (bin) / 0 (dec)	1101111 (bin) / 111 (dec)	111 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
0	0111001 (bin) / 57 (dec)	1110001 (bin) / 113 (dec)	0 (bin) / 0 (dec)	0101010 (bin) / 42 (dec)	42 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
31	0001110 (bin) / 14 (dec)	1001111 (bin) / 79 (dec)	1 (bin) / 1 (dec)	1011110 (bin) / 94 (dec)	94 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
23	0000010 (bin) / 2 (dec)	1000001 (bin) / 65 (dec)	1 (bin) / 1 (dec)	1000100 (bin) / 68 (dec)	68 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
27	1011100 (bin) / 92 (dec)	0101110 (bin) / 46 (dec)	1 (bin) / 1 (dec)	0001011 (bin) / 11 (dec)	11 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
30	0011100 (bin) / 28 (dec)	1000011 (bin) / 67 (dec)	1 (bin) / 1 (dec)	1100000 (bin) / 96 (dec)	96 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
33	0110001 (bin) / 49 (dec)	1001100 (bin) / 76 (dec)	0 (bin) / 0 (dec)	1111101 (bin) / 125 (dec)	125 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
19	1011000 (bin) / 88 (dec)	0110010 (bin) / 50 (dec)	1 (bin) / 1 (dec)	0001011 (bin) / 11 (dec)	11 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
18	1011110 (bin) / 94 (dec)	1110111 (bin) / 119 (dec)	0 (bin) / 0 (dec)	1010101 (bin) / 85 (dec)	85 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	0100100 (bin) / 36 (dec)	0101100 (bin) / 44 (dec)	0 (bin) / 0 (dec)	1010000 (bin) / 80 (dec)	80 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
21	1111001 (bin) / 121 (dec)	1111111 (bin) / 127 (dec)	1 (bin) / 1 (dec)	1111001 (bin) / 121 (dec)	121 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
4	1011100 (bin) / 92 (dec)	0000101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	1100001 (bin) / 97 (dec)	97 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
2	0010011 (bin) / 19 (dec)	1010010 (bin) / 82 (dec)	1 (bin) / 1 (dec)	1100110 (bin) / 102 (dec)	102 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
7	1001000 (bin) / 72 (dec)	1010110 (bin) / 86 (dec)	0 (bin) / 0 (dec)	0011110 (bin) / 30 (dec)	30 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
12	1001100 (bin) / 76 (dec)	0000101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	1010001 (bin) / 81 (dec)	81 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	1110101 (bin) / 117 (dec)	0000100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	1111001 (bin) / 121 (dec)	121 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
17	1000000 (bin) / 64 (dec)	1000110 (bin) / 70 (dec)	0 (bin) / 0 (dec)	0000110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
16	1011010 (bin) / 90 (dec)	0011010 (bin) / 26 (dec)	1 (bin) / 1 (dec)	1110101 (bin) / 117 (dec)	117 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
29	0001000 (bin) / 8 (dec)	0100010 (bin) / 34 (dec)	0 (bin) / 0 (dec)	0101010 (bin) / 42 (dec)	42 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
28	0001100 (bin) / 12 (dec)	0000011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	0001111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
20	1011100 (bin) / 92 (dec)	0101111 (bin) / 47 (dec)	0 (bin) / 0 (dec)	0001011 (bin) / 11 (dec)	11 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
22	1111000 (bin) / 120 (dec)	0111011 (bin) / 59 (dec)	1 (bin) / 1 (dec)	0110100 (bin) / 52 (dec)	52 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
15	1001110 (bin) / 78 (dec)	1001000 (bin) / 72 (dec)	0 (bin) / 0 (dec)	0010110 (bin) / 22 (dec)	22 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
24	1100101 (bin) / 101 (dec)	0111011 (bin) / 59 (dec)	0 (bin) / 0 (dec)	0100000 (bin) / 32 (dec)	32 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
14	0011010 (bin) / 26 (dec)	0101001 (bin) / 41 (dec)	0 (bin) / 0 (dec)	1000011 (bin) / 67 (dec)	67 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
def matches(self, filename):
    return self.pattern in filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << self.bit_width) - 1
    if "cin" in test_case:
        sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
        outs = {
            "sum": sum_val & max_val,
            "cout": sum_val >> self.bit_width
    }
else:
    sum_val = test_case["a"] + test_case["b"]
    outs = {
        "sum": sum_val & max_val,
        "cout": sum_val >> self.bit_width
    }
return outs
```

Testbench for ripple_carry_adder with parameter(s) N8

Total tests: 35

Passed tests: 35

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
10	00011000 (bin) / 24 (dec)	01010111 (bin) / 87 (dec)	0 (bin) / 0 (dec)	01101111 (bin) / 111 (dec)	111 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
29	01111101 (bin) / 125 (dec)	10100010 (bin) / 162 (dec)	1 (bin) / 1 (dec)	00100000 (bin) / 32 (dec)	32 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
15	00101001 (bin) / 41 (dec)	00000010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	00101011 (bin) / 43 (dec)	43 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
21	10011101 (bin) / 157 (dec)	00000010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	10100000 (bin) / 160 (dec)	160 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
12	10000011 (bin) / 131 (dec)	11001000 (bin) / 200 (dec)	1 (bin) / 1 (dec)	01001100 (bin) / 76 (dec)	76 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	01100011 (bin) / 99 (dec)	00111100 (bin) / 60 (dec)	0 (bin) / 0 (dec)	10011111 (bin) / 159 (dec)	159 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	00010100 (bin) / 20 (dec)	00010101 (bin) / 21 (dec)	0 (bin) / 0 (dec)	00101001 (bin) / 41 (dec)	41 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
26	11010100 (bin) / 212 (dec)	10110100 (bin) / 180 (dec)	0 (bin) / 0 (dec)	10001000 (bin) / 136 (dec)	136 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
19	11100110 (bin) / 230 (dec)	11011010 (bin) / 218 (dec)	1 (bin) / 1 (dec)	11000001 (bin) / 193 (dec)	193 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
28	01001111 (bin) / 79 (dec)	00110000 (bin) / 48 (dec)	0 (bin) / 0 (dec)	01111111 (bin) / 127 (dec)	127 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
23	01000001 (bin) / 65 (dec)	00100111 (bin) / 39 (dec)	0 (bin) / 0 (dec)	01101000 (bin) / 104 (dec)	104 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
14	10101101 (bin) / 173 (dec)	00111111 (bin) / 63 (dec)	1 (bin) / 1 (dec)	11101101 (bin) / 237 (dec)	237 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
5	01100011 (bin) / 99 (dec)	01001001 (bin) / 73 (dec)	0 (bin) / 0 (dec)	10101100 (bin) / 172 (dec)	172 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
0	00101101 (bin) / 45 (dec)	01000101 (bin) / 69 (dec)	1 (bin) / 1 (dec)	01110011 (bin) / 115 (dec)	115 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
32	11111110 (bin) / 254 (dec)	01010001 (bin) / 81 (dec)	0 (bin) / 0 (dec)	01001111 (bin) / 79 (dec)	79 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
11	01011011 (bin) / 91 (dec)	00011110 (bin) / 30 (dec)	0 (bin) / 0 (dec)	01111001 (bin) / 121 (dec)	121 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
34	01000111 (bin) / 71 (dec)	00111001 (bin) / 57 (dec)	0 (bin) / 0 (dec)	10000000 (bin) / 128 (dec)	128 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
16	00110111 (bin) / 55 (dec)	11111111 (bin) / 255 (dec)	1 (bin) / 1 (dec)	00110111 (bin) / 55 (dec)	55 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
22	10110001 (bin) / 177 (dec)	10011001 (bin) / 153 (dec)	0 (bin) / 0 (dec)	01001010 (bin) / 74 (dec)	74 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
2	00100101 (bin) / 37 (dec)	01011011 (bin) / 91 (dec)	0 (bin) / 0 (dec)	10000000 (bin) / 128 (dec)	128 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
9	00101101 (bin) / 45 (dec)	01110111 (bin) / 119 (dec)	1 (bin) / 1 (dec)	10100101 (bin) / 165 (dec)	165 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
33	01110000 (bin) / 112 (dec)	00111011 (bin) / 59 (dec)	1 (bin) / 1 (dec)	10101100 (bin) / 172 (dec)	172 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
17	00100100 (bin) / 36 (dec)	11001001 (bin) / 201 (dec)	1 (bin) / 1 (dec)	11101110 (bin) / 238 (dec)	238 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
31	10111011 (bin) / 187 (dec)	01100100 (bin) / 100 (dec)	1 (bin) / 1 (dec)	00100000 (bin) / 32 (dec)	32 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
4	01011010 (bin) / 90 (dec)	01010000 (bin) / 80 (dec)	1 (bin) / 1 (dec)	10101011 (bin) / 171 (dec)	171 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
def matches(self, filename):
    return self.pattern in filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << self.bit_width) - 1
    if "cin" in test_case:
        sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
        outs = {
            "sum": sum_val & max_val,
            "cout": sum_val >> self.bit_width
    }
else:
    sum_val = test_case["a"] + test_case["b"]
    outs = {
        "sum": sum_val & max_val,
        "cout": sum_val >> self.bit_width
    }
return outs
```

Testbench for full_adder with parameter(s)

Total tests: 8

Passed tests: 8

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
4	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
0	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
7	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
3	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
5	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
2	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
6	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
            def matches(self, filename):
                return self.pattern in filename
```

Generate expected values function:

```
            def generate_expected(self, test_case):
                max_val = (1 << self.bit_width) - 1
                if "cin" in test_case:
                    sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
                    outs = {
                        "sum": sum_val & max_val,
                        "cout": sum_val >> self.bit_width
                    }
                else:
                    sum_val = test_case["a"] + test_case["b"]
                    outs = {
                        "sum": sum_val & max_val,
                        "cout": sum_val >> self.bit_width
                    }
                return outs
```

Testbench for half_adder with parameter(s)

Total tests: 4

Passed tests: 4

Failed tests: 0

Test Case	Input a	Input b	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
0	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
2	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: AdderRule

Input Variables: a, b, cin

Output Variables: sum, cout

Bit Width: 8

Pattern: SubstringPattern

```
        def matches(self, filename):
            return self.pattern in filename
```

Generate expected values function:

```
        def generate_expected(self, test_case):
            max_val = (1 << self.bit_width) - 1
            if "cin" in test_case:
                sum_val = test_case["a"] + test_case["b"] + test_case["cin"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            else:
                sum_val = test_case["a"] + test_case["b"]
                outs = {
                    "sum": sum_val & max_val,
                    "cout": sum_val >> self.bit_width
                }
            return outs
```