Testbenching Report for carry_save_adder

Table of Contents

Testbench Summary

Component	Total Tests	Passed	Failed
carry_save_adder_N1	8	8	0
carry_save_adder_N2	64	64	0
carry_save_adder_N3	512	512	0
carry_save_adder_N4	1638	1638	0
carry_save_adder_N5	1639	1639	0
carry_save_adder_N6	1638	1638	0
carry_save_adder_N7	1639	1639	0
carry_save_adder_C_N1	8	8	0
carry_save_adder_C_N2	64	64	0
carry_save_adder_C_N3	512	512	0
carry_save_adder_C_N4	1639	1639	0
carry_save_adder_C_N5	1638	1638	0
carry_save_adder_C_N6	1639	1639	0
carry_save_adder_C_N7	1638	1638	0
carry_save_adder_C_N8	1639	1639	0
full_adder_	8	8	0
half_adder_	4	4	0
ripple_carry_adder_N1	8	8	0
ripple_carry_adder_N2	32	32	0
ripple_carry_adder_N3	128	128	0
ripple_carry_adder_N4	512	512	0
ripple_carry_adder_N5	1024	1024	0
ripple_carry_adder_N6	1638	1638	0
ripple_carry_adder_N7	1639	1639	0
ripple_carry_adder_N8	1638	1638	0

Testbench for carry_save_adder with parameter(s) N1

Total tests: 8

Passed tests: 8

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
1	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
6	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	Passed
0	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	Passed
4	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
7	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	Passed
5	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
2	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	Passed
3	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder with parameter(s) N2

Total tests: 64

Passed tests: 64

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
52	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
12	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
36	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
28	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	10 (bin) / 2 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
51	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
35	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	00 (bin) / 0 (dec)	001 (bin) / 1 (dec)	1 (dec)	Passed
20	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
7	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	011 (bin) / 3 (dec)	3 (dec)	Passed
49	00 (bin) / 0 (dec)	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
42	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	000 (bin) / 0 (dec)	0 (dec)	Passed
31	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	Passed
24	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	000 (bin) / 0 (dec)	0 (dec)	Passed
29	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
60	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
43	00 (bin) / 0 (dec)	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	011 (bin) / 3 (dec)	3 (dec)	Passed
63	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
58	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	001 (bin) / 1 (dec)	1 (dec)	Passed
26	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	111 (bin) / 7 (dec)	7 (dec)	Passed
62	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	00 (bin) / 0 (dec)	010 (bin) / 2 (dec)	2 (dec)	Passed
50	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
18	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	110 (bin) / 6 (dec)	6 (dec)	Passed
44	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
56	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
9	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	110 (bin) / 6 (dec)	6 (dec)	Passed
41	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	01 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder with parameter(s) N3

Total tests: 512

Passed tests: 512

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
150	100 (bin) / 4 (dec)	110 (bin) / 6 (dec)	001 (bin) / 1 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
169	100 (bin) / 4 (dec)	111 (bin) / 7 (dec)	000 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
200	101 (bin) / 5 (dec)	010 (bin) / 2 (dec)	100 (bin) / 4 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
496	101 (bin) / 5 (dec)	110 (bin) / 6 (dec)	000 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
333	101 (bin) / 5 (dec)	100 (bin) / 4 (dec)	111 (bin) / 7 (dec)	0000 (bin) / 0 (dec)	0 (dec)	Passed
335	100 (bin) / 4 (dec)	001 (bin) / 1 (dec)	110 (bin) / 6 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
374	101 (bin) / 5 (dec)	000 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0101 (bin) / 5 (dec)	5 (dec)	Passed
492	000 (bin) / 0 (dec)	101 (bin) / 5 (dec)	111 (bin) / 7 (dec)	1100 (bin) / 12 (dec)	12 (dec)	Passed
204	100 (bin) / 4 (dec)	001 (bin) / 1 (dec)	000 (bin) / 0 (dec)	0101 (bin) / 5 (dec)	5 (dec)	Passed
488	110 (bin) / 6 (dec)	101 (bin) / 5 (dec)	100 (bin) / 4 (dec)	1111 (bin) / 15 (dec)	15 (dec)	Passed
172	010 (bin) / 2 (dec)	000 (bin) / 0 (dec)	010 (bin) / 2 (dec)	0100 (bin) / 4 (dec)	4 (dec)	Passed
192	000 (bin) / 0 (dec)	000 (bin) / 0 (dec)	101 (bin) / 5 (dec)	0101 (bin) / 5 (dec)	5 (dec)	Passed
23	001 (bin) / 1 (dec)	010 (bin) / 2 (dec)	110 (bin) / 6 (dec)	1001 (bin) / 9 (dec)	9 (dec)	Passed
456	110 (bin) / 6 (dec)	101 (bin) / 5 (dec)	011 (bin) / 3 (dec)	1110 (bin) / 14 (dec)	14 (dec)	Passed
416	110 (bin) / 6 (dec)	010 (bin) / 2 (dec)	110 (bin) / 6 (dec)	1110 (bin) / 14 (dec)	14 (dec)	Passed
320	000 (bin) / 0 (dec)	111 (bin) / 7 (dec)	101 (bin) / 5 (dec)	1100 (bin) / 12 (dec)	12 (dec)	Passed
286	101 (bin) / 5 (dec)	101 (bin) / 5 (dec)	101 (bin) / 5 (dec)	1111 (bin) / 15 (dec)	15 (dec)	Passed
485	011 (bin) / 3 (dec)	000 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0011 (bin) / 3 (dec)	3 (dec)	Passed
474	101 (bin) / 5 (dec)	011 (bin) / 3 (dec)	100 (bin) / 4 (dec)	1100 (bin) / 12 (dec)	12 (dec)	Passed
244	001 (bin) / 1 (dec)	111 (bin) / 7 (dec)	010 (bin) / 2 (dec)	1010 (bin) / 10 (dec)	10 (dec)	Passed
463	111 (bin) / 7 (dec)	010 (bin) / 2 (dec)	111 (bin) / 7 (dec)	0000 (bin) / 0 (dec)	0 (dec)	Passed
362	100 (bin) / 4 (dec)	111 (bin) / 7 (dec)	111 (bin) / 7 (dec)	0010 (bin) / 2 (dec)	2 (dec)	Passed
263	001 (bin) / 1 (dec)	110 (bin) / 6 (dec)	111 (bin) / 7 (dec)	1110 (bin) / 14 (dec)	14 (dec)	Passed
392	000 (bin) / 0 (dec)	011 (bin) / 3 (dec)	110 (bin) / 6 (dec)	1001 (bin) / 9 (dec)	9 (dec)	Passed
324	010 (bin) / 2 (dec)	110 (bin) / 6 (dec)	011 (bin) / 3 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder with parameter(s) N4

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
1439	1101 (bin) / 13 (dec)	1100 (bin) / 12 (dec)	0101 (bin) / 5 (dec)	11110 (bin) / 30 (dec)	30 (dec)	Passed
652	1001 (bin) / 9 (dec)	0110 (bin) / 6 (dec)	0100 (bin) / 4 (dec)	10011 (bin) / 19 (dec)	19 (dec)	Passed
847	1001 (bin) / 9 (dec)	0010 (bin) / 2 (dec)	0011 (bin) / 3 (dec)	01110 (bin) / 14 (dec)	14 (dec)	Passed
454	0111 (bin) / 7 (dec)	1011 (bin) / 11 (dec)	0101 (bin) / 5 (dec)	10111 (bin) / 23 (dec)	23 (dec)	Passed
1389	1000 (bin) / 8 (dec)	1101 (bin) / 13 (dec)	0010 (bin) / 2 (dec)	10111 (bin) / 23 (dec)	23 (dec)	Passed
735	1000 (bin) / 8 (dec)	1111 (bin) / 15 (dec)	1000 (bin) / 8 (dec)	11111 (bin) / 31 (dec)	31 (dec)	Passed
1318	1001 (bin) / 9 (dec)	0000 (bin) / 0 (dec)	0010 (bin) / 2 (dec)	01011 (bin) / 11 (dec)	11 (dec)	Passed
1267	1101 (bin) / 13 (dec)	1001 (bin) / 9 (dec)	1111 (bin) / 15 (dec)	00101 (bin) / 5 (dec)	5 (dec)	Passed
1183	1000 (bin) / 8 (dec)	1011 (bin) / 11 (dec)	1111 (bin) / 15 (dec)	00010 (bin) / 2 (dec)	2 (dec)	Passed
849	1001 (bin) / 9 (dec)	1110 (bin) / 14 (dec)	1000 (bin) / 8 (dec)	11111 (bin) / 31 (dec)	31 (dec)	Passed
1463	1010 (bin) / 10 (dec)	1111 (bin) / 15 (dec)	1100 (bin) / 12 (dec)	00101 (bin) / 5 (dec)	5 (dec)	Passed
102	1101 (bin) / 13 (dec)	1110 (bin) / 14 (dec)	0011 (bin) / 3 (dec)	11110 (bin) / 30 (dec)	30 (dec)	Passed
719	1010 (bin) / 10 (dec)	1010 (bin) / 10 (dec)	1001 (bin) / 9 (dec)	11101 (bin) / 29 (dec)	29 (dec)	Passed
868	0000 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	1110 (bin) / 14 (dec)	11001 (bin) / 25 (dec)	25 (dec)	Passed
639	1101 (bin) / 13 (dec)	0101 (bin) / 5 (dec)	0100 (bin) / 4 (dec)	10110 (bin) / 22 (dec)	22 (dec)	Passed
1093	1000 (bin) / 8 (dec)	1011 (bin) / 11 (dec)	0100 (bin) / 4 (dec)	10111 (bin) / 23 (dec)	23 (dec)	Passed
413	1011 (bin) / 11 (dec)	1010 (bin) / 10 (dec)	0001 (bin) / 1 (dec)	10110 (bin) / 22 (dec)	22 (dec)	Passed
815	0110 (bin) / 6 (dec)	0000 (bin) / 0 (dec)	0000 (bin) / 0 (dec)	00110 (bin) / 6 (dec)	6 (dec)	Passed
357	0110 (bin) / 6 (dec)	1000 (bin) / 8 (dec)	1100 (bin) / 12 (dec)	11010 (bin) / 26 (dec)	26 (dec)	Passed
491	0000 (bin) / 0 (dec)	1100 (bin) / 12 (dec)	1100 (bin) / 12 (dec)	11000 (bin) / 24 (dec)	24 (dec)	Passed
309	1100 (bin) / 12 (dec)	0100 (bin) / 4 (dec)	0010 (bin) / 2 (dec)	10010 (bin) / 18 (dec)	18 (dec)	Passed
447	1110 (bin) / 14 (dec)	0000 (bin) / 0 (dec)	1000 (bin) / 8 (dec)	10110 (bin) / 22 (dec)	22 (dec)	Passed
1591	0101 (bin) / 5 (dec)	1001 (bin) / 9 (dec)	0010 (bin) / 2 (dec)	10000 (bin) / 16 (dec)	16 (dec)	Passed
1288	0100 (bin) / 4 (dec)	0111 (bin) / 7 (dec)	0110 (bin) / 6 (dec)	10001 (bin) / 17 (dec)	17 (dec)	Passed
93	1000 (bin) / 8 (dec)	0100 (bin) / 4 (dec)	0000 (bin) / 0 (dec)	01100 (bin) / 12 (dec)	12 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder with parameter(s) N5

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
34	10010 (bin) / 18 (dec)	00100 (bin) / 4 (dec)	10111 (bin) / 23 (dec)	101101 (bin) / 45 (dec)	45 (dec)	Passed
1481	01110 (bin) / 14 (dec)	00000 (bin) / 0 (dec)	11110 (bin) / 30 (dec)	101100 (bin) / 44 (dec)	44 (dec)	Passed
1439	00001 (bin) / 1 (dec)	11010 (bin) / 26 (dec)	10010 (bin) / 18 (dec)	101101 (bin) / 45 (dec)	45 (dec)	Passed
1065	01110 (bin) / 14 (dec)	10010 (bin) / 18 (dec)	11000 (bin) / 24 (dec)	111000 (bin) / 56 (dec)	56 (dec)	Passed
211	01100 (bin) / 12 (dec)	10001 (bin) / 17 (dec)	10001 (bin) / 17 (dec)	101110 (bin) / 46 (dec)	46 (dec)	Passed
458	10110 (bin) / 22 (dec)	01101 (bin) / 13 (dec)	10001 (bin) / 17 (dec)	110100 (bin) / 52 (dec)	52 (dec)	Passed
312	10000 (bin) / 16 (dec)	10011 (bin) / 19 (dec)	01010 (bin) / 10 (dec)	101101 (bin) / 45 (dec)	45 (dec)	Passed
1451	00101 (bin) / 5 (dec)	00000 (bin) / 0 (dec)	11111 (bin) / 31 (dec)	100100 (bin) / 36 (dec)	36 (dec)	Passed
189	00011 (bin) / 3 (dec)	11001 (bin) / 25 (dec)	00110 (bin) / 6 (dec)	100010 (bin) / 34 (dec)	34 (dec)	Passed
595	00111 (bin) / 7 (dec)	11011 (bin) / 27 (dec)	00110 (bin) / 6 (dec)	101000 (bin) / 40 (dec)	40 (dec)	Passed
251	10001 (bin) / 17 (dec)	10110 (bin) / 22 (dec)	00001 (bin) / 1 (dec)	101000 (bin) / 40 (dec)	40 (dec)	Passed
1530	11000 (bin) / 24 (dec)	00111 (bin) / 7 (dec)	00011 (bin) / 3 (dec)	100010 (bin) / 34 (dec)	34 (dec)	Passed
1006	11110 (bin) / 30 (dec)	11110 (bin) / 30 (dec)	10010 (bin) / 18 (dec)	001110 (bin) / 14 (dec)	14 (dec)	Passed
1027	00110 (bin) / 6 (dec)	00000 (bin) / 0 (dec)	11011 (bin) / 27 (dec)	100001 (bin) / 33 (dec)	33 (dec)	Passed
1550	00100 (bin) / 4 (dec)	00010 (bin) / 2 (dec)	00111 (bin) / 7 (dec)	001101 (bin) / 13 (dec)	13 (dec)	Passed
1238	01010 (bin) / 10 (dec)	01010 (bin) / 10 (dec)	10100 (bin) / 20 (dec)	101000 (bin) / 40 (dec)	40 (dec)	Passed
726	00010 (bin) / 2 (dec)	00100 (bin) / 4 (dec)	01100 (bin) / 12 (dec)	010010 (bin) / 18 (dec)	18 (dec)	Passed
101	10000 (bin) / 16 (dec)	01111 (bin) / 15 (dec)	11001 (bin) / 25 (dec)	111000 (bin) / 56 (dec)	56 (dec)	Passed
569	11000 (bin) / 24 (dec)	00111 (bin) / 7 (dec)	01110 (bin) / 14 (dec)	101101 (bin) / 45 (dec)	45 (dec)	Passed
474	11000 (bin) / 24 (dec)	00100 (bin) / 4 (dec)	11001 (bin) / 25 (dec)	110101 (bin) / 53 (dec)	53 (dec)	Passed
1283	10110 (bin) / 22 (dec)	01000 (bin) / 8 (dec)	01010 (bin) / 10 (dec)	101000 (bin) / 40 (dec)	40 (dec)	Passed
1295	01001 (bin) / 9 (dec)	01100 (bin) / 12 (dec)	00100 (bin) / 4 (dec)	011001 (bin) / 25 (dec)	25 (dec)	Passed
460	01000 (bin) / 8 (dec)	00001 (bin) / 1 (dec)	11011 (bin) / 27 (dec)	100100 (bin) / 36 (dec)	36 (dec)	Passed
149	11000 (bin) / 24 (dec)	01010 (bin) / 10 (dec)	10100 (bin) / 20 (dec)	110110 (bin) / 54 (dec)	54 (dec)	Passed
1173	10100 (bin) / 20 (dec)	01011 (bin) / 11 (dec)	01100 (bin) / 12 (dec)	101011 (bin) / 43 (dec)	43 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder with parameter(s) N6

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
952	100111 (bin) / 39 (dec)	010111 (bin) / 23 (dec)	101110 (bin) / 46 (dec)	1101100 (bin) / 108 (dec)	108 (dec)	Passed
136	111101 (bin) / 61 (dec)	110011 (bin) / 51 (dec)	100001 (bin) / 33 (dec)	0010001 (bin) / 17 (dec)	17 (dec)	Passed
407	001010 (bin) / 10 (dec)	100001 (bin) / 33 (dec)	000111 (bin) / 7 (dec)	0110010 (bin) / 50 (dec)	50 (dec)	Passed
497	010010 (bin) / 18 (dec)	101111 (bin) / 47 (dec)	100101 (bin) / 37 (dec)	1100110 (bin) / 102 (dec)	102 (dec)	Passed
904	000110 (bin) / 6 (dec)	100100 (bin) / 36 (dec)	000000 (bin) / 0 (dec)	0101010 (bin) / 42 (dec)	42 (dec)	Passed
246	010011 (bin) / 19 (dec)	011101 (bin) / 29 (dec)	001110 (bin) / 14 (dec)	0111110 (bin) / 62 (dec)	62 (dec)	Passed
608	111011 (bin) / 59 (dec)	000011 (bin) / 3 (dec)	111101 (bin) / 61 (dec)	1111011 (bin) / 123 (dec)	123 (dec)	Passed
843	111010 (bin) / 58 (dec)	011000 (bin) / 24 (dec)	000001 (bin) / 1 (dec)	1010011 (bin) / 83 (dec)	83 (dec)	Passed
1455	001100 (bin) / 12 (dec)	000011 (bin) / 3 (dec)	101011 (bin) / 43 (dec)	0111010 (bin) / 58 (dec)	58 (dec)	Passed
1472	011110 (bin) / 30 (dec)	011111 (bin) / 31 (dec)	010110 (bin) / 22 (dec)	1010011 (bin) / 83 (dec)	83 (dec)	Passed
1439	000000 (bin) / 0 (dec)	101111 (bin) / 47 (dec)	011010 (bin) / 26 (dec)	1001001 (bin) / 73 (dec)	73 (dec)	Passed
768	000100 (bin) / 4 (dec)	100010 (bin) / 34 (dec)	001111 (bin) / 15 (dec)	0110101 (bin) / 53 (dec)	53 (dec)	Passed
550	110001 (bin) / 49 (dec)	111011 (bin) / 59 (dec)	111110 (bin) / 62 (dec)	0101010 (bin) / 42 (dec)	42 (dec)	Passed
175	100000 (bin) / 32 (dec)	011100 (bin) / 28 (dec)	010111 (bin) / 23 (dec)	1010011 (bin) / 83 (dec)	83 (dec)	Passed
278	101011 (bin) / 43 (dec)	100100 (bin) / 36 (dec)	111110 (bin) / 62 (dec)	0001101 (bin) / 13 (dec)	13 (dec)	Passed
1266	011010 (bin) / 26 (dec)	110000 (bin) / 48 (dec)	110011 (bin) / 51 (dec)	1111101 (bin) / 125 (dec)	125 (dec)	Passed
198	110110 (bin) / 54 (dec)	000100 (bin) / 4 (dec)	010101 (bin) / 21 (dec)	1001111 (bin) / 79 (dec)	79 (dec)	Passed
1384	010111 (bin) / 23 (dec)	000011 (bin) / 3 (dec)	011001 (bin) / 25 (dec)	0110011 (bin) / 51 (dec)	51 (dec)	Passed
579	011010 (bin) / 26 (dec)	000000 (bin) / 0 (dec)	011111 (bin) / 31 (dec)	0111001 (bin) / 57 (dec)	57 (dec)	Passed
242	100000 (bin) / 32 (dec)	101110 (bin) / 46 (dec)	111100 (bin) / 60 (dec)	0001010 (bin) / 10 (dec)	10 (dec)	Passed
1636	000110 (bin) / 6 (dec)	000111 (bin) / 7 (dec)	110111 (bin) / 55 (dec)	1000100 (bin) / 68 (dec)	68 (dec)	Passed
1039	001011 (bin) / 11 (dec)	011100 (bin) / 28 (dec)	101100 (bin) / 44 (dec)	1010011 (bin) / 83 (dec)	83 (dec)	Passed
339	000001 (bin) / 1 (dec)	100100 (bin) / 36 (dec)	101000 (bin) / 40 (dec)	1001101 (bin) / 77 (dec)	77 (dec)	Passed
1190	010001 (bin) / 17 (dec)	011001 (bin) / 25 (dec)	010011 (bin) / 19 (dec)	0111101 (bin) / 61 (dec)	61 (dec)	Passed
1550	111100 (bin) / 60 (dec)	011111 (bin) / 31 (dec)	010100 (bin) / 20 (dec)	1101111 (bin) / 111 (dec)	111 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder with parameter(s) N7

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
575	1111110 (bin) / 126 (dec)	1110110 (bin) / 118 (dec)	0001000 (bin) / 8 (dec)	11111100 (bin) / 252 (dec)	252 (dec)	Passed
1019	1010100 (bin) / 84 (dec)	1110000 (bin) / 112 (dec)	0001101 (bin) / 13 (dec)	11010001 (bin) / 209 (dec)	209 (dec)	Passed
555	0010010 (bin) / 18 (dec)	0001001 (bin) / 9 (dec)	0001010 (bin) / 10 (dec)	00100101 (bin) / 37 (dec)	37 (dec)	Passed
123	0000101 (bin) / 5 (dec)	0100011 (bin) / 35 (dec)	0100100 (bin) / 36 (dec)	01001100 (bin) / 76 (dec)	76 (dec)	Passed
997	1011001 (bin) / 89 (dec)	0101100 (bin) / 44 (dec)	1111011 (bin) / 123 (dec)	00000000 (bin) / 0 (dec)	0 (dec)	Passed
29	0011010 (bin) / 26 (dec)	0000101 (bin) / 5 (dec)	1001000 (bin) / 72 (dec)	01100111 (bin) / 103 (dec)	103 (dec)	Passed
659	0101100 (bin) / 44 (dec)	0011000 (bin) / 24 (dec)	0001001 (bin) / 9 (dec)	01001101 (bin) / 77 (dec)	77 (dec)	Passed
1033	1011111 (bin) / 95 (dec)	0011011 (bin) / 27 (dec)	0001110 (bin) / 14 (dec)	10001000 (bin) / 136 (dec)	136 (dec)	Passed
624	1001010 (bin) / 74 (dec)	0000000 (bin) / 0 (dec)	1010010 (bin) / 82 (dec)	10011100 (bin) / 156 (dec)	156 (dec)	Passed
35	1100000 (bin) / 96 (dec)	1010010 (bin) / 82 (dec)	1001101 (bin) / 77 (dec)	11111111 (bin) / 255 (dec)	255 (dec)	Passed
1617	1001000 (bin) / 72 (dec)	1111101 (bin) / 125 (dec)	1001001 (bin) / 73 (dec)	00001110 (bin) / 14 (dec)	14 (dec)	Passed
1067	0100110 (bin) / 38 (dec)	0100111 (bin) / 39 (dec)	1100000 (bin) / 96 (dec)	10101101 (bin) / 173 (dec)	173 (dec)	Passed
1340	1111010 (bin) / 122 (dec)	1000111 (bin) / 71 (dec)	1110000 (bin) / 112 (dec)	00110001 (bin) / 49 (dec)	49 (dec)	Passed
196	1010001 (bin) / 81 (dec)	0011101 (bin) / 29 (dec)	0011110 (bin) / 30 (dec)	10001100 (bin) / 140 (dec)	140 (dec)	Passed
1203	0111111 (bin) / 63 (dec)	1001111 (bin) / 79 (dec)	1001010 (bin) / 74 (dec)	11011000 (bin) / 216 (dec)	216 (dec)	Passed
364	0100100 (bin) / 36 (dec)	1001100 (bin) / 76 (dec)	1000111 (bin) / 71 (dec)	10110111 (bin) / 183 (dec)	183 (dec)	Passed
476	0001011 (bin) / 11 (dec)	0000001 (bin) / 1 (dec)	0110010 (bin) / 50 (dec)	00111110 (bin) / 62 (dec)	62 (dec)	Passed
1542	1100001 (bin) / 97 (dec)	0011001 (bin) / 25 (dec)	1011000 (bin) / 88 (dec)	11010010 (bin) / 210 (dec)	210 (dec)	Passed
709	0010100 (bin) / 20 (dec)	1100100 (bin) / 100 (dec)	0111000 (bin) / 56 (dec)	10110000 (bin) / 176 (dec)	176 (dec)	Passed
589	0101010 (bin) / 42 (dec)	1100100 (bin) / 100 (dec)	0100110 (bin) / 38 (dec)	10110100 (bin) / 180 (dec)	180 (dec)	Passed
828	0000111 (bin) / 7 (dec)	1111011 (bin) / 123 (dec)	1111110 (bin) / 126 (dec)	00000000 (bin) / 0 (dec)	0 (dec)	Passed
1304	1100111 (bin) / 103 (dec)	0110101 (bin) / 53 (dec)	0000001 (bin) / 1 (dec)	10011101 (bin) / 157 (dec)	157 (dec)	Passed
1632	0100001 (bin) / 33 (dec)	0100011 (bin) / 35 (dec)	0001110 (bin) / 14 (dec)	01010010 (bin) / 82 (dec)	82 (dec)	Passed
933	0010111 (bin) / 23 (dec)	0100101 (bin) / 37 (dec)	1010110 (bin) / 86 (dec)	10010010 (bin) / 146 (dec)	146 (dec)	Passed
184	1101000 (bin) / 104 (dec)	1000010 (bin) / 66 (dec)	1010010 (bin) / 82 (dec)	11111100 (bin) / 252 (dec)	252 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder with parameter(s) N8

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output result (Actual)	Expected result	Status
1263	10010010 (bin) / 146 (dec)	00101010 (bin) / 42 (dec)	01011000 (bin) / 88 (dec)	100010100 (bin) / 276 (dec)	276 (dec)	Passed
59	00110110 (bin) / 54 (dec)	10001010 (bin) / 138 (dec)	10110101 (bin) / 181 (dec)	101110101 (bin) / 373 (dec)	373 (dec)	Passed
42	01011011 (bin) / 91 (dec)	10100010 (bin) / 162 (dec)	00000110 (bin) / 6 (dec)	100000011 (bin) / 259 (dec)	259 (dec)	Passed
1523	00010111 (bin) / 23 (dec)	10101100 (bin) / 172 (dec)	10100111 (bin) / 167 (dec)	101101010 (bin) / 362 (dec)	362 (dec)	Passed
430	11000101 (bin) / 197 (dec)	10011110 (bin) / 158 (dec)	00000101 (bin) / 5 (dec)	101101000 (bin) / 360 (dec)	360 (dec)	Passed
374	00011000 (bin) / 24 (dec)	10001011 (bin) / 139 (dec)	10001001 (bin) / 137 (dec)	100101100 (bin) / 300 (dec)	300 (dec)	Passed
87	11011001 (bin) / 217 (dec)	10111001 (bin) / 185 (dec)	10100110 (bin) / 166 (dec)	000111000 (bin) / 56 (dec)	56 (dec)	Passed
97	10000000 (bin) / 128 (dec)	10110111 (bin) / 183 (dec)	11010101 (bin) / 213 (dec)	000001100 (bin) / 12 (dec)	12 (dec)	Passed
1148	11001010 (bin) / 202 (dec)	00010100 (bin) / 20 (dec)	00100011 (bin) / 35 (dec)	100000001 (bin) / 257 (dec)	257 (dec)	Passed
912	01100000 (bin) / 96 (dec)	00000100 (bin) / 4 (dec)	10001100 (bin) / 140 (dec)	011110000 (bin) / 240 (dec)	240 (dec)	Passed
1502	10010101 (bin) / 149 (dec)	01000001 (bin) / 65 (dec)	10000001 (bin) / 129 (dec)	101010111 (bin) / 343 (dec)	343 (dec)	Passed
843	01111001 (bin) / 121 (dec)	00100100 (bin) / 36 (dec)	11011100 (bin) / 220 (dec)	101111001 (bin) / 377 (dec)	377 (dec)	Passed
736	00011001 (bin) / 25 (dec)	10111110 (bin) / 190 (dec)	00110000 (bin) / 48 (dec)	100000111 (bin) / 263 (dec)	263 (dec)	Passed
901	10101101 (bin) / 173 (dec)	01110011 (bin) / 115 (dec)	01001010 (bin) / 74 (dec)	101101010 (bin) / 362 (dec)	362 (dec)	Passed
945	01101000 (bin) / 104 (dec)	11001000 (bin) / 200 (dec)	10101110 (bin) / 174 (dec)	111011110 (bin) / 478 (dec)	478 (dec)	Passed
419	10110101 (bin) / 181 (dec)	01110010 (bin) / 114 (dec)	11100011 (bin) / 227 (dec)	000001010 (bin) / 10 (dec)	10 (dec)	Passed
1086	11100111 (bin) / 231 (dec)	10011011 (bin) / 155 (dec)	10101111 (bin) / 175 (dec)	000110001 (bin) / 49 (dec)	49 (dec)	Passed
1466	00111001 (bin) / 57 (dec)	11000111 (bin) / 199 (dec)	00111001 (bin) / 57 (dec)	100111001 (bin) / 313 (dec)	313 (dec)	Passed
215	01010100 (bin) / 84 (dec)	01100110 (bin) / 102 (dec)	10100111 (bin) / 167 (dec)	101100001 (bin) / 353 (dec)	353 (dec)	Passed
1041	01110001 (bin) / 113 (dec)	10100010 (bin) / 162 (dec)	11101101 (bin) / 237 (dec)	000000000 (bin) / 0 (dec)	0 (dec)	Passed
277	00100100 (bin) / 36 (dec)	10000000 (bin) / 128 (dec)	10010000 (bin) / 144 (dec)	100110100 (bin) / 308 (dec)	308 (dec)	Passed
715	01101101 (bin) / 109 (dec)	11111001 (bin) / 249 (dec)	01110101 (bin) / 117 (dec)	111011011 (bin) / 475 (dec)	475 (dec)	Passed
1636	00011101 (bin) / 29 (dec)	00010011 (bin) / 19 (dec)	11011011 (bin) / 219 (dec)	100001011 (bin) / 267 (dec)	267 (dec)	Passed
835	11000000 (bin) / 192 (dec)	10110111 (bin) / 183 (dec)	10110101 (bin) / 181 (dec)	000101100 (bin) / 44 (dec)	44 (dec)	Passed
141	10011111 (bin) / 159 (dec)	10010010 (bin) / 146 (dec)	00101010 (bin) / 42 (dec)	101011011 (bin) / 347 (dec)	347 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N1

Total tests: 8

Passed tests: 8

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
6	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	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	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
3	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
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
5	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
7	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: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N2

Total tests: 64

Passed tests: 64

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
29	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	10 (bin) / 2 (dec)	2 (dec)	Passed
49	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
56	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
25	00 (bin) / 0 (dec)	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	3 (dec)	00 (bin) / 0 (dec)	0 (dec)	Passed
63	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
14	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	2 (dec)	10 (bin) / 2 (dec)	2 (dec)	Passed
38	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	0 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
55	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	2 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
12	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
0	00 (bin) / 0 (dec)	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
19	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	0 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
7	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	3 (dec)	00 (bin) / 0 (dec)	0 (dec)	Passed
35	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	0 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
23	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	1 (dec)	10 (bin) / 2 (dec)	2 (dec)	Passed
3	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	1 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
36	10 (bin) / 2 (dec)	00 (bin) / 0 (dec)	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	1 (dec)	10 (bin) / 2 (dec)	2 (dec)	Passed
50	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	2 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
8	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	0 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
41	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
62	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	1 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
59	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
45	00 (bin) / 0 (dec)	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	2 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
4	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	2 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed
61	11 (bin) / 3 (dec)	11 (bin) / 3 (dec)	10 (bin) / 2 (dec)	10 (bin) / 2 (dec)	2 (dec)	11 (bin) / 3 (dec)	3 (dec)	Passed
54	11 (bin) / 3 (dec)	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	01 (bin) / 1 (dec)	1 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N3

Total tests: 512

Passed tests: 512

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
129	000 (bin) / 0 (dec)	010 (bin) / 2 (dec)	001 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	000 (bin) / 0 (dec)	0 (dec)	Passed
311	111 (bin) / 7 (dec)	111 (bin) / 7 (dec)	011 (bin) / 3 (dec)	011 (bin) / 3 (dec)	3 (dec)	111 (bin) / 7 (dec)	7 (dec)	Passed
440	101 (bin) / 5 (dec)	001 (bin) / 1 (dec)	101 (bin) / 5 (dec)	001 (bin) / 1 (dec)	1 (dec)	101 (bin) / 5 (dec)	5 (dec)	Passed
387	010 (bin) / 2 (dec)	000 (bin) / 0 (dec)	111 (bin) / 7 (dec)	101 (bin) / 5 (dec)	5 (dec)	010 (bin) / 2 (dec)	2 (dec)	Passed
18	010 (bin) / 2 (dec)	000 (bin) / 0 (dec)	010 (bin) / 2 (dec)	000 (bin) / 0 (dec)	0 (dec)	010 (bin) / 2 (dec)	2 (dec)	Passed
54	111 (bin) / 7 (dec)	000 (bin) / 0 (dec)	001 (bin) / 1 (dec)	110 (bin) / 6 (dec)	6 (dec)	001 (bin) / 1 (dec)	1 (dec)	Passed
487	100 (bin) / 4 (dec)	100 (bin) / 4 (dec)	011 (bin) / 3 (dec)	011 (bin) / 3 (dec)	3 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
86	111 (bin) / 7 (dec)	011 (bin) / 3 (dec)	001 (bin) / 1 (dec)	101 (bin) / 5 (dec)	5 (dec)	011 (bin) / 3 (dec)	3 (dec)	Passed
280	010 (bin) / 2 (dec)	111 (bin) / 7 (dec)	110 (bin) / 6 (dec)	011 (bin) / 3 (dec)	3 (dec)	110 (bin) / 6 (dec)	6 (dec)	Passed
55	101 (bin) / 5 (dec)	000 (bin) / 0 (dec)	001 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	001 (bin) / 1 (dec)	1 (dec)	Passed
351	111 (bin) / 7 (dec)	011 (bin) / 3 (dec)	010 (bin) / 2 (dec)	110 (bin) / 6 (dec)	6 (dec)	011 (bin) / 3 (dec)	3 (dec)	Passed
296	001 (bin) / 1 (dec)	000 (bin) / 0 (dec)	010 (bin) / 2 (dec)	011 (bin) / 3 (dec)	3 (dec)	000 (bin) / 0 (dec)	0 (dec)	Passed
37	101 (bin) / 5 (dec)	011 (bin) / 3 (dec)	001 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	001 (bin) / 1 (dec)	1 (dec)	Passed
411	010 (bin) / 2 (dec)	011 (bin) / 3 (dec)	000 (bin) / 0 (dec)	001 (bin) / 1 (dec)	1 (dec)	010 (bin) / 2 (dec)	2 (dec)	Passed
260	010 (bin) / 2 (dec)	000 (bin) / 0 (dec)	000 (bin) / 0 (dec)	010 (bin) / 2 (dec)	2 (dec)	000 (bin) / 0 (dec)	0 (dec)	Passed
506	011 (bin) / 3 (dec)	100 (bin) / 4 (dec)	111 (bin) / 7 (dec)	000 (bin) / 0 (dec)	0 (dec)	111 (bin) / 7 (dec)	7 (dec)	Passed
279	110 (bin) / 6 (dec)	101 (bin) / 5 (dec)	100 (bin) / 4 (dec)	111 (bin) / 7 (dec)	7 (dec)	100 (bin) / 4 (dec)	4 (dec)	Passed
62	001 (bin) / 1 (dec)	000 (bin) / 0 (dec)	111 (bin) / 7 (dec)	110 (bin) / 6 (dec)	6 (dec)	001 (bin) / 1 (dec)	1 (dec)	Passed
270	101 (bin) / 5 (dec)	110 (bin) / 6 (dec)	110 (bin) / 6 (dec)	101 (bin) / 5 (dec)	5 (dec)	110 (bin) / 6 (dec)	6 (dec)	Passed
69	110 (bin) / 6 (dec)	010 (bin) / 2 (dec)	100 (bin) / 4 (dec)	000 (bin) / 0 (dec)	0 (dec)	110 (bin) / 6 (dec)	6 (dec)	Passed
272	110 (bin) / 6 (dec)	011 (bin) / 3 (dec)	011 (bin) / 3 (dec)	110 (bin) / 6 (dec)	6 (dec)	011 (bin) / 3 (dec)	3 (dec)	Passed
82	110 (bin) / 6 (dec)	111 (bin) / 7 (dec)	110 (bin) / 6 (dec)	111 (bin) / 7 (dec)	7 (dec)	110 (bin) / 6 (dec)	6 (dec)	Passed
207	011 (bin) / 3 (dec)	010 (bin) / 2 (dec)	100 (bin) / 4 (dec)	101 (bin) / 5 (dec)	5 (dec)	010 (bin) / 2 (dec)	2 (dec)	Passed
246	111 (bin) / 7 (dec)	101 (bin) / 5 (dec)	110 (bin) / 6 (dec)	100 (bin) / 4 (dec)	4 (dec)	111 (bin) / 7 (dec)	7 (dec)	Passed
184	011 (bin) / 3 (dec)	010 (bin) / 2 (dec)	110 (bin) / 6 (dec)	111 (bin) / 7 (dec)	7 (dec)	010 (bin) / 2 (dec)	2 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N4

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
1551	0011 (bin) / 3 (dec)	0111 (bin) / 7 (dec)	1110 (bin) / 14 (dec)	1010 (bin) / 10 (dec)	10 (dec)	0111 (bin) / 7 (dec)	7 (dec)	Passed
298	1101 (bin) / 13 (dec)	0010 (bin) / 2 (dec)	0110 (bin) / 6 (dec)	1001 (bin) / 9 (dec)	9 (dec)	0110 (bin) / 6 (dec)	6 (dec)	Passed
1399	0001 (bin) / 1 (dec)	1110 (bin) / 14 (dec)	0011 (bin) / 3 (dec)	1100 (bin) / 12 (dec)	12 (dec)	0011 (bin) / 3 (dec)	3 (dec)	Passed
422	1111 (bin) / 15 (dec)	0011 (bin) / 3 (dec)	0001 (bin) / 1 (dec)	1101 (bin) / 13 (dec)	13 (dec)	0011 (bin) / 3 (dec)	3 (dec)	Passed
458	1110 (bin) / 14 (dec)	1101 (bin) / 13 (dec)	0011 (bin) / 3 (dec)	0000 (bin) / 0 (dec)	0 (dec)	1111 (bin) / 15 (dec)	15 (dec)	Passed
575	0011 (bin) / 3 (dec)	1011 (bin) / 11 (dec)	0100 (bin) / 4 (dec)	1100 (bin) / 12 (dec)	12 (dec)	0011 (bin) / 3 (dec)	3 (dec)	Passed
1006	1101 (bin) / 13 (dec)	0100 (bin) / 4 (dec)	0100 (bin) / 4 (dec)	1101 (bin) / 13 (dec)	13 (dec)	0100 (bin) / 4 (dec)	4 (dec)	Passed
1447	1101 (bin) / 13 (dec)	1000 (bin) / 8 (dec)	1101 (bin) / 13 (dec)	1000 (bin) / 8 (dec)	8 (dec)	1101 (bin) / 13 (dec)	13 (dec)	Passed
1209	1111 (bin) / 15 (dec)	0001 (bin) / 1 (dec)	1011 (bin) / 11 (dec)	0101 (bin) / 5 (dec)	5 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
472	0010 (bin) / 2 (dec)	0100 (bin) / 4 (dec)	1110 (bin) / 14 (dec)	1000 (bin) / 8 (dec)	8 (dec)	0110 (bin) / 6 (dec)	6 (dec)	Passed
1015	1010 (bin) / 10 (dec)	1101 (bin) / 13 (dec)	1011 (bin) / 11 (dec)	1100 (bin) / 12 (dec)	12 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
474	0010 (bin) / 2 (dec)	1110 (bin) / 14 (dec)	0110 (bin) / 6 (dec)	1010 (bin) / 10 (dec)	10 (dec)	0110 (bin) / 6 (dec)	6 (dec)	Passed
1144	1001 (bin) / 9 (dec)	0100 (bin) / 4 (dec)	0101 (bin) / 5 (dec)	1000 (bin) / 8 (dec)	8 (dec)	0101 (bin) / 5 (dec)	5 (dec)	Passed
1276	1011 (bin) / 11 (dec)	1011 (bin) / 11 (dec)	1001 (bin) / 9 (dec)	1001 (bin) / 9 (dec)	9 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
1386	1001 (bin) / 9 (dec)	1111 (bin) / 15 (dec)	1001 (bin) / 9 (dec)	1111 (bin) / 15 (dec)	15 (dec)	1001 (bin) / 9 (dec)	9 (dec)	Passed
174	0100 (bin) / 4 (dec)	1011 (bin) / 11 (dec)	0000 (bin) / 0 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0000 (bin) / 0 (dec)	0 (dec)	Passed
283	1011 (bin) / 11 (dec)	0011 (bin) / 3 (dec)	1100 (bin) / 12 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1011 (bin) / 11 (dec)	11 (dec)	Passed
287	1001 (bin) / 9 (dec)	1000 (bin) / 8 (dec)	1110 (bin) / 14 (dec)	1111 (bin) / 15 (dec)	15 (dec)	1000 (bin) / 8 (dec)	8 (dec)	Passed
459	1100 (bin) / 12 (dec)	1000 (bin) / 8 (dec)	1110 (bin) / 14 (dec)	1010 (bin) / 10 (dec)	10 (dec)	1100 (bin) / 12 (dec)	12 (dec)	Passed
1285	1100 (bin) / 12 (dec)	0101 (bin) / 5 (dec)	0011 (bin) / 3 (dec)	1010 (bin) / 10 (dec)	10 (dec)	0101 (bin) / 5 (dec)	5 (dec)	Passed
1425	0100 (bin) / 4 (dec)	1001 (bin) / 9 (dec)	1101 (bin) / 13 (dec)	0000 (bin) / 0 (dec)	0 (dec)	1101 (bin) / 13 (dec)	13 (dec)	Passed
815	1001 (bin) / 9 (dec)	0100 (bin) / 4 (dec)	1000 (bin) / 8 (dec)	0101 (bin) / 5 (dec)	5 (dec)	1000 (bin) / 8 (dec)	8 (dec)	Passed
296	1100 (bin) / 12 (dec)	0111 (bin) / 7 (dec)	1100 (bin) / 12 (dec)	0111 (bin) / 7 (dec)	7 (dec)	1100 (bin) / 12 (dec)	12 (dec)	Passed
121	0000 (bin) / 0 (dec)	1000 (bin) / 8 (dec)	0001 (bin) / 1 (dec)	1001 (bin) / 9 (dec)	9 (dec)	0000 (bin) / 0 (dec)	0 (dec)	Passed
367	1001 (bin) / 9 (dec)	1001 (bin) / 9 (dec)	1001 (bin) / 9 (dec)	1001 (bin) / 9 (dec)	9 (dec)	1001 (bin) / 9 (dec)	9 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N5

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
1534	10101 (bin) / 21 (dec)	11011 (bin) / 27 (dec)	11010 (bin) / 26 (dec)	10100 (bin) / 20 (dec)	20 (dec)	11011 (bin) / 27 (dec)	27 (dec)	Passed
1544	00110 (bin) / 6 (dec)	10010 (bin) / 18 (dec)	11100 (bin) / 28 (dec)	01000 (bin) / 8 (dec)	8 (dec)	10110 (bin) / 22 (dec)	22 (dec)	Passed
246	11000 (bin) / 24 (dec)	11000 (bin) / 24 (dec)	10111 (bin) / 23 (dec)	10111 (bin) / 23 (dec)	23 (dec)	11000 (bin) / 24 (dec)	24 (dec)	Passed
1130	10011 (bin) / 19 (dec)	01011 (bin) / 11 (dec)	10011 (bin) / 19 (dec)	01011 (bin) / 11 (dec)	11 (dec)	10011 (bin) / 19 (dec)	19 (dec)	Passed
537	01001 (bin) / 9 (dec)	10111 (bin) / 23 (dec)	10011 (bin) / 19 (dec)	01101 (bin) / 13 (dec)	13 (dec)	10011 (bin) / 19 (dec)	19 (dec)	Passed
789	10100 (bin) / 20 (dec)	10101 (bin) / 21 (dec)	01001 (bin) / 9 (dec)	01000 (bin) / 8 (dec)	8 (dec)	10101 (bin) / 21 (dec)	21 (dec)	Passed
1378	01100 (bin) / 12 (dec)	00010 (bin) / 2 (dec)	01111 (bin) / 15 (dec)	00001 (bin) / 1 (dec)	1 (dec)	01110 (bin) / 14 (dec)	14 (dec)	Passed
930	10100 (bin) / 20 (dec)	00010 (bin) / 2 (dec)	01111 (bin) / 15 (dec)	11001 (bin) / 25 (dec)	25 (dec)	00110 (bin) / 6 (dec)	6 (dec)	Passed
1414	11010 (bin) / 26 (dec)	10010 (bin) / 18 (dec)	01101 (bin) / 13 (dec)	00101 (bin) / 5 (dec)	5 (dec)	11010 (bin) / 26 (dec)	26 (dec)	Passed
452	01100 (bin) / 12 (dec)	11111 (bin) / 31 (dec)	00110 (bin) / 6 (dec)	10101 (bin) / 21 (dec)	21 (dec)	01110 (bin) / 14 (dec)	14 (dec)	Passed
890	01010 (bin) / 10 (dec)	10100 (bin) / 20 (dec)	11010 (bin) / 26 (dec)	00100 (bin) / 4 (dec)	4 (dec)	11010 (bin) / 26 (dec)	26 (dec)	Passed
1584	00010 (bin) / 2 (dec)	11101 (bin) / 29 (dec)	01011 (bin) / 11 (dec)	10100 (bin) / 20 (dec)	20 (dec)	01011 (bin) / 11 (dec)	11 (dec)	Passed
375	10011 (bin) / 19 (dec)	00111 (bin) / 7 (dec)	01001 (bin) / 9 (dec)	11101 (bin) / 29 (dec)	29 (dec)	00011 (bin) / 3 (dec)	3 (dec)	Passed
1078	00101 (bin) / 5 (dec)	00111 (bin) / 7 (dec)	10100 (bin) / 20 (dec)	10110 (bin) / 22 (dec)	22 (dec)	00101 (bin) / 5 (dec)	5 (dec)	Passed
1126	01001 (bin) / 9 (dec)	00111 (bin) / 7 (dec)	01000 (bin) / 8 (dec)	00110 (bin) / 6 (dec)	6 (dec)	01001 (bin) / 9 (dec)	9 (dec)	Passed
431	00011 (bin) / 3 (dec)	01101 (bin) / 13 (dec)	10010 (bin) / 18 (dec)	11100 (bin) / 28 (dec)	28 (dec)	00011 (bin) / 3 (dec)	3 (dec)	Passed
1156	11101 (bin) / 29 (dec)	10011 (bin) / 19 (dec)	00001 (bin) / 1 (dec)	01111 (bin) / 15 (dec)	15 (dec)	10001 (bin) / 17 (dec)	17 (dec)	Passed
587	00010 (bin) / 2 (dec)	00101 (bin) / 5 (dec)	01000 (bin) / 8 (dec)	01111 (bin) / 15 (dec)	15 (dec)	00000 (bin) / 0 (dec)	0 (dec)	Passed
132	10001 (bin) / 17 (dec)	01100 (bin) / 12 (dec)	11000 (bin) / 24 (dec)	00101 (bin) / 5 (dec)	5 (dec)	11000 (bin) / 24 (dec)	24 (dec)	Passed
1503	10010 (bin) / 18 (dec)	00111 (bin) / 7 (dec)	00100 (bin) / 4 (dec)	10001 (bin) / 17 (dec)	17 (dec)	00110 (bin) / 6 (dec)	6 (dec)	Passed
773	10110 (bin) / 22 (dec)	00111 (bin) / 7 (dec)	10101 (bin) / 21 (dec)	00100 (bin) / 4 (dec)	4 (dec)	10111 (bin) / 23 (dec)	23 (dec)	Passed
1342	11101 (bin) / 29 (dec)	11011 (bin) / 27 (dec)	10001 (bin) / 17 (dec)	10111 (bin) / 23 (dec)	23 (dec)	11001 (bin) / 25 (dec)	25 (dec)	Passed
57	00101 (bin) / 5 (dec)	11010 (bin) / 26 (dec)	10001 (bin) / 17 (dec)	01110 (bin) / 14 (dec)	14 (dec)	10001 (bin) / 17 (dec)	17 (dec)	Passed
728	10011 (bin) / 19 (dec)	00110 (bin) / 6 (dec)	11100 (bin) / 28 (dec)	01001 (bin) / 9 (dec)	9 (dec)	10110 (bin) / 22 (dec)	22 (dec)	Passed
1033	01010 (bin) / 10 (dec)	01000 (bin) / 8 (dec)	00101 (bin) / 5 (dec)	00111 (bin) / 7 (dec)	7 (dec)	01000 (bin) / 8 (dec)	8 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N6

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
953	011011 (bin) / 27 (dec)	010001 (bin) / 17 (dec)	110000 (bin) / 48 (dec)	111010 (bin) / 58 (dec)	58 (dec)	010001 (bin) / 17 (dec)	17 (dec)	Passed
1102	100011 (bin) / 35 (dec)	111010 (bin) / 58 (dec)	001000 (bin) / 8 (dec)	010001 (bin) / 17 (dec)	17 (dec)	101010 (bin) / 42 (dec)	42 (dec)	Passed
1048	011111 (bin) / 31 (dec)	110001 (bin) / 49 (dec)	110010 (bin) / 50 (dec)	011100 (bin) / 28 (dec)	28 (dec)	110011 (bin) / 51 (dec)	51 (dec)	Passed
1376	011110 (bin) / 30 (dec)	100011 (bin) / 35 (dec)	100001 (bin) / 33 (dec)	011100 (bin) / 28 (dec)	28 (dec)	100011 (bin) / 35 (dec)	35 (dec)	Passed
39	100100 (bin) / 36 (dec)	111100 (bin) / 60 (dec)	101001 (bin) / 41 (dec)	110001 (bin) / 49 (dec)	49 (dec)	101100 (bin) / 44 (dec)	44 (dec)	Passed
361	111000 (bin) / 56 (dec)	001000 (bin) / 8 (dec)	110101 (bin) / 53 (dec)	000101 (bin) / 5 (dec)	5 (dec)	111000 (bin) / 56 (dec)	56 (dec)	Passed
834	010011 (bin) / 19 (dec)	110111 (bin) / 55 (dec)	110110 (bin) / 54 (dec)	010010 (bin) / 18 (dec)	18 (dec)	110111 (bin) / 55 (dec)	55 (dec)	Passed
1541	111101 (bin) / 61 (dec)	000101 (bin) / 5 (dec)	110001 (bin) / 49 (dec)	001001 (bin) / 9 (dec)	9 (dec)	110101 (bin) / 53 (dec)	53 (dec)	Passed
1419	001111 (bin) / 15 (dec)	001101 (bin) / 13 (dec)	010001 (bin) / 17 (dec)	010011 (bin) / 19 (dec)	19 (dec)	001101 (bin) / 13 (dec)	13 (dec)	Passed
850	100110 (bin) / 38 (dec)	100100 (bin) / 36 (dec)	010011 (bin) / 19 (dec)	010001 (bin) / 17 (dec)	17 (dec)	100110 (bin) / 38 (dec)	38 (dec)	Passed
473	001100 (bin) / 12 (dec)	001111 (bin) / 15 (dec)	110110 (bin) / 54 (dec)	110101 (bin) / 53 (dec)	53 (dec)	001110 (bin) / 14 (dec)	14 (dec)	Passed
292	111100 (bin) / 60 (dec)	000100 (bin) / 4 (dec)	100110 (bin) / 38 (dec)	011110 (bin) / 30 (dec)	30 (dec)	100100 (bin) / 36 (dec)	36 (dec)	Passed
684	000000 (bin) / 0 (dec)	100010 (bin) / 34 (dec)	000101 (bin) / 5 (dec)	100111 (bin) / 39 (dec)	39 (dec)	000000 (bin) / 0 (dec)	0 (dec)	Passed
1008	000110 (bin) / 6 (dec)	101011 (bin) / 43 (dec)	101000 (bin) / 40 (dec)	000101 (bin) / 5 (dec)	5 (dec)	101010 (bin) / 42 (dec)	42 (dec)	Passed
440	010011 (bin) / 19 (dec)	000100 (bin) / 4 (dec)	111010 (bin) / 58 (dec)	101101 (bin) / 45 (dec)	45 (dec)	010010 (bin) / 18 (dec)	18 (dec)	Passed
461	111111 (bin) / 63 (dec)	011100 (bin) / 28 (dec)	010011 (bin) / 19 (dec)	110000 (bin) / 48 (dec)	48 (dec)	011111 (bin) / 31 (dec)	31 (dec)	Passed
180	011010 (bin) / 26 (dec)	110101 (bin) / 53 (dec)	110110 (bin) / 54 (dec)	011001 (bin) / 25 (dec)	25 (dec)	110110 (bin) / 54 (dec)	54 (dec)	Passed
1519	001010 (bin) / 10 (dec)	111000 (bin) / 56 (dec)	100011 (bin) / 35 (dec)	010001 (bin) / 17 (dec)	17 (dec)	101010 (bin) / 42 (dec)	42 (dec)	Passed
1369	010100 (bin) / 20 (dec)	110110 (bin) / 54 (dec)	001110 (bin) / 14 (dec)	101100 (bin) / 44 (dec)	44 (dec)	010110 (bin) / 22 (dec)	22 (dec)	Passed
415	010110 (bin) / 22 (dec)	000000 (bin) / 0 (dec)	100011 (bin) / 35 (dec)	110101 (bin) / 53 (dec)	53 (dec)	000010 (bin) / 2 (dec)	2 (dec)	Passed
628	110111 (bin) / 55 (dec)	101101 (bin) / 45 (dec)	100001 (bin) / 33 (dec)	111011 (bin) / 59 (dec)	59 (dec)	100101 (bin) / 37 (dec)	37 (dec)	Passed
285	001101 (bin) / 13 (dec)	011011 (bin) / 27 (dec)	111000 (bin) / 56 (dec)	101110 (bin) / 46 (dec)	46 (dec)	011001 (bin) / 25 (dec)	25 (dec)	Passed
149	011011 (bin) / 27 (dec)	000111 (bin) / 7 (dec)	100011 (bin) / 35 (dec)	111111 (bin) / 63 (dec)	63 (dec)	000011 (bin) / 3 (dec)	3 (dec)	Passed
194	010110 (bin) / 22 (dec)	100011 (bin) / 35 (dec)	000011 (bin) / 3 (dec)	110110 (bin) / 54 (dec)	54 (dec)	000011 (bin) / 3 (dec)	3 (dec)	Passed
1011	011001 (bin) / 25 (dec)	000100 (bin) / 4 (dec)	111001 (bin) / 57 (dec)	100100 (bin) / 36 (dec)	36 (dec)	011001 (bin) / 25 (dec)	25 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N7

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
1049	1000000 (bin) / 64 (dec)	1011111 (bin) / 95 (dec)	1111001 (bin) / 121 (dec)	1100110 (bin) / 102 (dec)	102 (dec)	1011001 (bin) / 89 (dec)	89 (dec)	Passed
346	0001001 (bin) / 9 (dec)	1111000 (bin) / 120 (dec)	0000011 (bin) / 3 (dec)	1110010 (bin) / 114 (dec)	114 (dec)	0001001 (bin) / 9 (dec)	9 (dec)	Passed
1349	1000001 (bin) / 65 (dec)	0111000 (bin) / 56 (dec)	1100101 (bin) / 101 (dec)	0011100 (bin) / 28 (dec)	28 (dec)	1100001 (bin) / 97 (dec)	97 (dec)	Passed
981	1100101 (bin) / 101 (dec)	0010010 (bin) / 18 (dec)	1111101 (bin) / 125 (dec)	0001010 (bin) / 10 (dec)	10 (dec)	1110101 (bin) / 117 (dec)	117 (dec)	Passed
1385	0001100 (bin) / 12 (dec)	1011110 (bin) / 94 (dec)	0111001 (bin) / 57 (dec)	1101011 (bin) / 107 (dec)	107 (dec)	0011100 (bin) / 28 (dec)	28 (dec)	Passed
848	1011101 (bin) / 93 (dec)	0101010 (bin) / 42 (dec)	0010010 (bin) / 18 (dec)	1100101 (bin) / 101 (dec)	101 (dec)	0011010 (bin) / 26 (dec)	26 (dec)	Passed
829	0011111 (bin) / 31 (dec)	1001101 (bin) / 77 (dec)	1111001 (bin) / 121 (dec)	0101011 (bin) / 43 (dec)	43 (dec)	1011101 (bin) / 93 (dec)	93 (dec)	Passed
1516	0000111 (bin) / 7 (dec)	0111100 (bin) / 60 (dec)	0100000 (bin) / 32 (dec)	0011011 (bin) / 27 (dec)	27 (dec)	0100100 (bin) / 36 (dec)	36 (dec)	Passed
907	0111100 (bin) / 60 (dec)	1011101 (bin) / 93 (dec)	0101000 (bin) / 40 (dec)	1001001 (bin) / 73 (dec)	73 (dec)	0111100 (bin) / 60 (dec)	60 (dec)	Passed
881	1110111 (bin) / 119 (dec)	0101010 (bin) / 42 (dec)	1001011 (bin) / 75 (dec)	0010110 (bin) / 22 (dec)	22 (dec)	1101011 (bin) / 107 (dec)	107 (dec)	Passed
709	0110011 (bin) / 51 (dec)	0111011 (bin) / 59 (dec)	0111001 (bin) / 57 (dec)	0110001 (bin) / 49 (dec)	49 (dec)	0111011 (bin) / 59 (dec)	59 (dec)	Passed
376	1100111 (bin) / 103 (dec)	1011101 (bin) / 93 (dec)	1100110 (bin) / 102 (dec)	1011100 (bin) / 92 (dec)	92 (dec)	1100111 (bin) / 103 (dec)	103 (dec)	Passed
182	1011001 (bin) / 89 (dec)	1100100 (bin) / 100 (dec)	1110111 (bin) / 119 (dec)	1001010 (bin) / 74 (dec)	74 (dec)	1110101 (bin) / 117 (dec)	117 (dec)	Passed
1203	1100001 (bin) / 97 (dec)	1101101 (bin) / 109 (dec)	0110010 (bin) / 50 (dec)	0111110 (bin) / 62 (dec)	62 (dec)	1100001 (bin) / 97 (dec)	97 (dec)	Passed
766	1000011 (bin) / 67 (dec)	1000010 (bin) / 66 (dec)	0011000 (bin) / 24 (dec)	0011001 (bin) / 25 (dec)	25 (dec)	1000010 (bin) / 66 (dec)	66 (dec)	Passed
1058	0011010 (bin) / 26 (dec)	0111100 (bin) / 60 (dec)	0111111 (bin) / 63 (dec)	0011001 (bin) / 25 (dec)	25 (dec)	0111110 (bin) / 62 (dec)	62 (dec)	Passed
972	1101100 (bin) / 108 (dec)	0110100 (bin) / 52 (dec)	1010110 (bin) / 86 (dec)	0001110 (bin) / 14 (dec)	14 (dec)	1110100 (bin) / 116 (dec)	116 (dec)	Passed
1069	0100000 (bin) / 32 (dec)	1110101 (bin) / 117 (dec)	1111001 (bin) / 121 (dec)	0101100 (bin) / 44 (dec)	44 (dec)	1110001 (bin) / 113 (dec)	113 (dec)	Passed
131	1101000 (bin) / 104 (dec)	1111111 (bin) / 127 (dec)	0111111 (bin) / 63 (dec)	0101000 (bin) / 40 (dec)	40 (dec)	1111111 (bin) / 127 (dec)	127 (dec)	Passed
1239	0001111 (bin) / 15 (dec)	1110010 (bin) / 114 (dec)	1011000 (bin) / 88 (dec)	0100101 (bin) / 37 (dec)	37 (dec)	1011010 (bin) / 90 (dec)	90 (dec)	Passed
614	1110010 (bin) / 114 (dec)	0100111 (bin) / 39 (dec)	0101100 (bin) / 44 (dec)	1111001 (bin) / 121 (dec)	121 (dec)	0100110 (bin) / 38 (dec)	38 (dec)	Passed
85	0011010 (bin) / 26 (dec)	1100101 (bin) / 101 (dec)	1000000 (bin) / 64 (dec)	0111111 (bin) / 63 (dec)	63 (dec)	1000000 (bin) / 64 (dec)	64 (dec)	Passed
584	0000101 (bin) / 5 (dec)	1001110 (bin) / 78 (dec)	0011101 (bin) / 29 (dec)	1010110 (bin) / 86 (dec)	86 (dec)	0001101 (bin) / 13 (dec)	13 (dec)	Passed
1557	1011001 (bin) / 89 (dec)	1101000 (bin) / 104 (dec)	1100111 (bin) / 103 (dec)	1010110 (bin) / 86 (dec)	86 (dec)	1101001 (bin) / 105 (dec)	105 (dec)	Passed
1036	1010100 (bin) / 84 (dec)	0010111 (bin) / 23 (dec)	1111111 (bin) / 127 (dec)	0111100 (bin) / 60 (dec)	60 (dec)	1010111 (bin) / 87 (dec)	87 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

Testbench for carry_save_adder_l2 with parameter(s) N8

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input c	Output sum (Actual)	Expected sum	Output carry (Actual)	Expected carry	Status
441	11000001 (bin) / 193 (dec)	11011101 (bin) / 221 (dec)	10100001 (bin) / 161 (dec)	10111101 (bin) / 189 (dec)	189 (dec)	11000001 (bin) / 193 (dec)	193 (dec)	Passed
1002	01101101 (bin) / 109 (dec)	11111111 (bin) / 255 (dec)	00011111 (bin) / 31 (dec)	10001101 (bin) / 141 (dec)	141 (dec)	01111111 (bin) / 127 (dec)	127 (dec)	Passed
771	01000101 (bin) / 69 (dec)	01001000 (bin) / 72 (dec)	00010110 (bin) / 22 (dec)	00011011 (bin) / 27 (dec)	27 (dec)	01000100 (bin) / 68 (dec)	68 (dec)	Passed
1029	00110111 (bin) / 55 (dec)	10111111 (bin) / 191 (dec)	00101111 (bin) / 47 (dec)	10100111 (bin) / 167 (dec)	167 (dec)	00111111 (bin) / 63 (dec)	63 (dec)	Passed
1080	11110010 (bin) / 242 (dec)	10101101 (bin) / 173 (dec)	00100010 (bin) / 34 (dec)	01111101 (bin) / 125 (dec)	125 (dec)	10100010 (bin) / 162 (dec)	162 (dec)	Passed
956	10000111 (bin) / 135 (dec)	01010011 (bin) / 83 (dec)	01111110 (bin) / 126 (dec)	10101010 (bin) / 170 (dec)	170 (dec)	01010111 (bin) / 87 (dec)	87 (dec)	Passed
982	00111011 (bin) / 59 (dec)	10000010 (bin) / 130 (dec)	00001001 (bin) / 9 (dec)	10110000 (bin) / 176 (dec)	176 (dec)	00001011 (bin) / 11 (dec)	11 (dec)	Passed
371	00111110 (bin) / 62 (dec)	11100001 (bin) / 225 (dec)	11010011 (bin) / 211 (dec)	00001100 (bin) / 12 (dec)	12 (dec)	11110011 (bin) / 243 (dec)	243 (dec)	Passed
986	10111100 (bin) / 188 (dec)	00011111 (bin) / 31 (dec)	11110010 (bin) / 242 (dec)	01010001 (bin) / 81 (dec)	81 (dec)	10111110 (bin) / 190 (dec)	190 (dec)	Passed
594	11100100 (bin) / 228 (dec)	00001001 (bin) / 9 (dec)	10101010 (bin) / 170 (dec)	01000111 (bin) / 71 (dec)	71 (dec)	10101000 (bin) / 168 (dec)	168 (dec)	Passed
1552	11101100 (bin) / 236 (dec)	01100100 (bin) / 100 (dec)	00111001 (bin) / 57 (dec)	10110001 (bin) / 177 (dec)	177 (dec)	01101100 (bin) / 108 (dec)	108 (dec)	Passed
1413	11100110 (bin) / 230 (dec)	11101111 (bin) / 239 (dec)	11100001 (bin) / 225 (dec)	11101000 (bin) / 232 (dec)	232 (dec)	11100111 (bin) / 231 (dec)	231 (dec)	Passed
85	11010001 (bin) / 209 (dec)	01011011 (bin) / 91 (dec)	01011000 (bin) / 88 (dec)	11010010 (bin) / 210 (dec)	210 (dec)	01011001 (bin) / 89 (dec)	89 (dec)	Passed
1147	00111111 (bin) / 63 (dec)	01100011 (bin) / 99 (dec)	01111110 (bin) / 126 (dec)	00100010 (bin) / 34 (dec)	34 (dec)	01111111 (bin) / 127 (dec)	127 (dec)	Passed
588	10010011 (bin) / 147 (dec)	11011010 (bin) / 218 (dec)	01010111 (bin) / 87 (dec)	00011110 (bin) / 30 (dec)	30 (dec)	11010011 (bin) / 211 (dec)	211 (dec)	Passed
874	10010101 (bin) / 149 (dec)	01110110 (bin) / 118 (dec)	00000011 (bin) / 3 (dec)	11100000 (bin) / 224 (dec)	224 (dec)	00010111 (bin) / 23 (dec)	23 (dec)	Passed
535	10110000 (bin) / 176 (dec)	11101000 (bin) / 232 (dec)	00110110 (bin) / 54 (dec)	01101110 (bin) / 110 (dec)	110 (dec)	10110000 (bin) / 176 (dec)	176 (dec)	Passed
292	11100110 (bin) / 230 (dec)	11010100 (bin) / 212 (dec)	01100010 (bin) / 98 (dec)	01010000 (bin) / 80 (dec)	80 (dec)	11100110 (bin) / 230 (dec)	230 (dec)	Passed
1151	00100000 (bin) / 32 (dec)	11011001 (bin) / 217 (dec)	01011111 (bin) / 95 (dec)	10100110 (bin) / 166 (dec)	166 (dec)	01011001 (bin) / 89 (dec)	89 (dec)	Passed
637	11011010 (bin) / 218 (dec)	11111110 (bin) / 254 (dec)	10101011 (bin) / 171 (dec)	10001111 (bin) / 143 (dec)	143 (dec)	11111010 (bin) / 250 (dec)	250 (dec)	Passed
687	00001001 (bin) / 9 (dec)	01111111 (bin) / 127 (dec)	10000001 (bin) / 129 (dec)	11110111 (bin) / 247 (dec)	247 (dec)	00001001 (bin) / 9 (dec)	9 (dec)	Passed
187	01000011 (bin) / 67 (dec)	11100111 (bin) / 231 (dec)	10000000 (bin) / 128 (dec)	00100100 (bin) / 36 (dec)	36 (dec)	11000011 (bin) / 195 (dec)	195 (dec)	Passed
1242	01001111 (bin) / 79 (dec)	01110110 (bin) / 118 (dec)	00011011 (bin) / 27 (dec)	00100010 (bin) / 34 (dec)	34 (dec)	01011111 (bin) / 95 (dec)	95 (dec)	Passed
1214	01100110 (bin) / 102 (dec)	01011011 (bin) / 91 (dec)	01101010 (bin) / 106 (dec)	01010111 (bin) / 87 (dec)	87 (dec)	01101010 (bin) / 106 (dec)	106 (dec)	Passed
548	01000111 (bin) / 71 (dec)	11101000 (bin) / 232 (dec)	11111011 (bin) / 251 (dec)	01010100 (bin) / 84 (dec)	84 (dec)	11101011 (bin) / 235 (dec)	235 (dec)	Passed

Rule: CarrySaveAdderRule

Input Variables: a, b, c

Output Variables: result

Bit Width: 8

Pattern: StringMatchPattern

```
def matches(self, filename):
    #print(self.pattern, filename)
    return self.pattern == filename
```

Generate expected values function:

```
def generate_expected(self, test_case):
    max_val = (1 << (self.bit_width + 1)) - 1
    sum1 = (test_case["a"] ^ test_case["b"] ^ test_case["c"]) & max_val
    carry1 = ((test_case["a"] & test_case["b"]) | (test_case["b"] & test_case["c"]) | (test_case["c"] & test_case["a"])) << 1
    result = (sum1 + carry1) & max_val
    return {
        "result": result
    }
```

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
2	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
1	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
3	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
7	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
6	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
0	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

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
1	0 (bin) / 0 (dec)	1 (bin) / 1 (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 (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
0	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
```

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
0	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)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	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
7	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
4	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
6	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
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

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
6	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
11	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
25	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
21	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
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
7	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
19	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
0	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
12	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
22	10 (bin) / 2 (dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
13	11 (bin) / 3 (dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
14	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
16	01 (bin) / 1 (dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
24	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
15	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
23	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
5	10 (bin) / 2 (dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1	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
10	01 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
3	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
9	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
8	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
27	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
30	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	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

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: 128

Passed tests: 128

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
22	001 (bin) / 1 (dec)	111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
24	011 (bin) / 3 (dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
63	110 (bin) / 6 (dec)	001 (bin) / 1 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
15	000 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
69	000 (bin) / 0 (dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	010 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
89	000 (bin) / 0 (dec)	111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
61	010 (bin) / 2 (dec)	000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
76	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
77	101 (bin) / 5 (dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
26	011 (bin) / 3 (dec)	011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
99	111 (bin) / 7 (dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
55	001 (bin) / 1 (dec)	010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
67	101 (bin) / 5 (dec)	001 (bin) / 1 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
100	100 (bin) / 4 (dec)	000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
4	101 (bin) / 5 (dec)	000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
85	100 (bin) / 4 (dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
43	100 (bin) / 4 (dec)	001 (bin) / 1 (dec)	1 (bin) / 1 (dec)	110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
108	111 (bin) / 7 (dec)	001 (bin) / 1 (dec)	0 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
66	111 (bin) / 7 (dec)	111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
96	110 (bin) / 6 (dec)	011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
7	101 (bin) / 5 (dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
31	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
3	101 (bin) / 5 (dec)	100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
90	101 (bin) / 5 (dec)	101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
32	001 (bin) / 1 (dec)	111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	001 (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) N4

Total tests: 512

Passed tests: 512

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
369	0101 (bin) / 5 (dec)	0000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
246	0001 (bin) / 1 (dec)	0100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	0110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
125	0100 (bin) / 4 (dec)	0111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
379	1111 (bin) / 15 (dec)	0111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	0110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
334	0011 (bin) / 3 (dec)	0000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
75	0001 (bin) / 1 (dec)	0111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	1001 (bin) / 9 (dec)	9 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
493	1011 (bin) / 11 (dec)	1010 (bin) / 10 (dec)	1 (bin) / 1 (dec)	0110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
308	1010 (bin) / 10 (dec)	1101 (bin) / 13 (dec)	1 (bin) / 1 (dec)	1000 (bin) / 8 (dec)	8 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
330	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
65	1111 (bin) / 15 (dec)	0000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
145	0011 (bin) / 3 (dec)	1101 (bin) / 13 (dec)	0 (bin) / 0 (dec)	0000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
50	1101 (bin) / 13 (dec)	0011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	0001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
230	1110 (bin) / 14 (dec)	0101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	0011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
120	1101 (bin) / 13 (dec)	1000 (bin) / 8 (dec)	0 (bin) / 0 (dec)	0101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
356	0101 (bin) / 5 (dec)	1000 (bin) / 8 (dec)	1 (bin) / 1 (dec)	1110 (bin) / 14 (dec)	14 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
15	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
503	0011 (bin) / 3 (dec)	1011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
17	1111 (bin) / 15 (dec)	1111 (bin) / 15 (dec)	1 (bin) / 1 (dec)	1111 (bin) / 15 (dec)	15 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
68	0001 (bin) / 1 (dec)	1100 (bin) / 12 (dec)	0 (bin) / 0 (dec)	1101 (bin) / 13 (dec)	13 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
235	1011 (bin) / 11 (dec)	1011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	0111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	0000 (bin) / 0 (dec)	1111 (bin) / 15 (dec)	1 (bin) / 1 (dec)	0000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
94	0011 (bin) / 3 (dec)	1100 (bin) / 12 (dec)	0 (bin) / 0 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
324	1010 (bin) / 10 (dec)	1100 (bin) / 12 (dec)	1 (bin) / 1 (dec)	0111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
273	0100 (bin) / 4 (dec)	1001 (bin) / 9 (dec)	0 (bin) / 0 (dec)	1101 (bin) / 13 (dec)	13 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
123	0110 (bin) / 6 (dec)	1010 (bin) / 10 (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: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
215	11000 (bin) / 24 (dec)	01011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	00100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
66	00110 (bin) / 6 (dec)	00001 (bin) / 1 (dec)	1 (bin) / 1 (dec)	01000 (bin) / 8 (dec)	8 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1342	10111 (bin) / 23 (dec)	01110 (bin) / 14 (dec)	1 (bin) / 1 (dec)	00110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1502	11101 (bin) / 29 (dec)	00011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	00001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
239	00010 (bin) / 2 (dec)	11110 (bin) / 30 (dec)	0 (bin) / 0 (dec)	00000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1603	11010 (bin) / 26 (dec)	00111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	00010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1105	10001 (bin) / 17 (dec)	10000 (bin) / 16 (dec)	0 (bin) / 0 (dec)	00001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1598	10100 (bin) / 20 (dec)	11111 (bin) / 31 (dec)	1 (bin) / 1 (dec)	10100 (bin) / 20 (dec)	20 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1198	10110 (bin) / 22 (dec)	10000 (bin) / 16 (dec)	0 (bin) / 0 (dec)	00110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1293	10110 (bin) / 22 (dec)	00110 (bin) / 6 (dec)	0 (bin) / 0 (dec)	11100 (bin) / 28 (dec)	28 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1188	01011 (bin) / 11 (dec)	11000 (bin) / 24 (dec)	1 (bin) / 1 (dec)	00100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
119	01001 (bin) / 9 (dec)	11011 (bin) / 27 (dec)	1 (bin) / 1 (dec)	00101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
553	01001 (bin) / 9 (dec)	01011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	10101 (bin) / 21 (dec)	21 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
477	01110 (bin) / 14 (dec)	11101 (bin) / 29 (dec)	1 (bin) / 1 (dec)	01100 (bin) / 12 (dec)	12 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1383	11011 (bin) / 27 (dec)	11110 (bin) / 30 (dec)	0 (bin) / 0 (dec)	11001 (bin) / 25 (dec)	25 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1549	00100 (bin) / 4 (dec)	00111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	01011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
206	11101 (bin) / 29 (dec)	11101 (bin) / 29 (dec)	0 (bin) / 0 (dec)	11010 (bin) / 26 (dec)	26 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1216	01001 (bin) / 9 (dec)	10101 (bin) / 21 (dec)	1 (bin) / 1 (dec)	11111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1544	11011 (bin) / 27 (dec)	01000 (bin) / 8 (dec)	1 (bin) / 1 (dec)	00100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
144	10011 (bin) / 19 (dec)	11110 (bin) / 30 (dec)	1 (bin) / 1 (dec)	10010 (bin) / 18 (dec)	18 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
57	10011 (bin) / 19 (dec)	10100 (bin) / 20 (dec)	1 (bin) / 1 (dec)	01000 (bin) / 8 (dec)	8 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1199	01001 (bin) / 9 (dec)	10101 (bin) / 21 (dec)	0 (bin) / 0 (dec)	11110 (bin) / 30 (dec)	30 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
934	11111 (bin) / 31 (dec)	11000 (bin) / 24 (dec)	1 (bin) / 1 (dec)	11000 (bin) / 24 (dec)	24 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1604	00001 (bin) / 1 (dec)	11111 (bin) / 31 (dec)	0 (bin) / 0 (dec)	00000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1143	01100 (bin) / 12 (dec)	10110 (bin) / 22 (dec)	0 (bin) / 0 (dec)	00010 (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) N6

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
1078	101101 (bin) / 45 (dec)	000111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	110100 (bin) / 52 (dec)	52 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
199	110111 (bin) / 55 (dec)	111000 (bin) / 56 (dec)	1 (bin) / 1 (dec)	110000 (bin) / 48 (dec)	48 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
232	010111 (bin) / 23 (dec)	100001 (bin) / 33 (dec)	1 (bin) / 1 (dec)	111001 (bin) / 57 (dec)	57 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
867	011100 (bin) / 28 (dec)	101101 (bin) / 45 (dec)	0 (bin) / 0 (dec)	001001 (bin) / 9 (dec)	9 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1337	000110 (bin) / 6 (dec)	101110 (bin) / 46 (dec)	0 (bin) / 0 (dec)	110100 (bin) / 52 (dec)	52 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
423	101001 (bin) / 41 (dec)	101111 (bin) / 47 (dec)	0 (bin) / 0 (dec)	011000 (bin) / 24 (dec)	24 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
69	111111 (bin) / 63 (dec)	100111 (bin) / 39 (dec)	1 (bin) / 1 (dec)	100111 (bin) / 39 (dec)	39 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
509	110001 (bin) / 49 (dec)	111000 (bin) / 56 (dec)	0 (bin) / 0 (dec)	101001 (bin) / 41 (dec)	41 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
465	111101 (bin) / 61 (dec)	110001 (bin) / 49 (dec)	0 (bin) / 0 (dec)	101110 (bin) / 46 (dec)	46 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1148	101110 (bin) / 46 (dec)	110110 (bin) / 54 (dec)	1 (bin) / 1 (dec)	100101 (bin) / 37 (dec)	37 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
370	010011 (bin) / 19 (dec)	100111 (bin) / 39 (dec)	0 (bin) / 0 (dec)	111010 (bin) / 58 (dec)	58 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1117	101101 (bin) / 45 (dec)	001110 (bin) / 14 (dec)	1 (bin) / 1 (dec)	111100 (bin) / 60 (dec)	60 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1002	010010 (bin) / 18 (dec)	101011 (bin) / 43 (dec)	1 (bin) / 1 (dec)	111110 (bin) / 62 (dec)	62 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1171	100100 (bin) / 36 (dec)	110110 (bin) / 54 (dec)	0 (bin) / 0 (dec)	011010 (bin) / 26 (dec)	26 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1575	110001 (bin) / 49 (dec)	000101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	110111 (bin) / 55 (dec)	55 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1194	010100 (bin) / 20 (dec)	011110 (bin) / 30 (dec)	1 (bin) / 1 (dec)	110011 (bin) / 51 (dec)	51 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1335	101111 (bin) / 47 (dec)	101110 (bin) / 46 (dec)	1 (bin) / 1 (dec)	011110 (bin) / 30 (dec)	30 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1595	110100 (bin) / 52 (dec)	110101 (bin) / 53 (dec)	1 (bin) / 1 (dec)	101010 (bin) / 42 (dec)	42 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
706	000101 (bin) / 5 (dec)	010111 (bin) / 23 (dec)	0 (bin) / 0 (dec)	011100 (bin) / 28 (dec)	28 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1258	111100 (bin) / 60 (dec)	001101 (bin) / 13 (dec)	1 (bin) / 1 (dec)	001010 (bin) / 10 (dec)	10 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1323	101111 (bin) / 47 (dec)	100000 (bin) / 32 (dec)	0 (bin) / 0 (dec)	001111 (bin) / 15 (dec)	15 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
503	111010 (bin) / 58 (dec)	001010 (bin) / 10 (dec)	0 (bin) / 0 (dec)	000100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
992	011111 (bin) / 31 (dec)	011101 (bin) / 29 (dec)	0 (bin) / 0 (dec)	111100 (bin) / 60 (dec)	60 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
627	000100 (bin) / 4 (dec)	011101 (bin) / 29 (dec)	0 (bin) / 0 (dec)	100001 (bin) / 33 (dec)	33 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1221	000001 (bin) / 1 (dec)	010001 (bin) / 17 (dec)	1 (bin) / 1 (dec)	010011 (bin) / 19 (dec)	19 (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: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
463	1011010 (bin) / 90 (dec)	0000101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	1100000 (bin) / 96 (dec)	96 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1396	0111000 (bin) / 56 (dec)	0111101 (bin) / 61 (dec)	1 (bin) / 1 (dec)	1110110 (bin) / 118 (dec)	118 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
242	0010111 (bin) / 23 (dec)	1001101 (bin) / 77 (dec)	0 (bin) / 0 (dec)	1100100 (bin) / 100 (dec)	100 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1128	0110111 (bin) / 55 (dec)	0010011 (bin) / 19 (dec)	0 (bin) / 0 (dec)	1001010 (bin) / 74 (dec)	74 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
410	1010111 (bin) / 87 (dec)	0110001 (bin) / 49 (dec)	0 (bin) / 0 (dec)	0001000 (bin) / 8 (dec)	8 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
166	0111010 (bin) / 58 (dec)	0011110 (bin) / 30 (dec)	1 (bin) / 1 (dec)	1011001 (bin) / 89 (dec)	89 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1425	0110110 (bin) / 54 (dec)	0000000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0110110 (bin) / 54 (dec)	54 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
198	1000010 (bin) / 66 (dec)	0111100 (bin) / 60 (dec)	0 (bin) / 0 (dec)	1111110 (bin) / 126 (dec)	126 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
447	0010000 (bin) / 16 (dec)	0100001 (bin) / 33 (dec)	1 (bin) / 1 (dec)	0110010 (bin) / 50 (dec)	50 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
918	1011011 (bin) / 91 (dec)	0011011 (bin) / 27 (dec)	1 (bin) / 1 (dec)	1110111 (bin) / 119 (dec)	119 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1476	1101100 (bin) / 108 (dec)	0111011 (bin) / 59 (dec)	0 (bin) / 0 (dec)	0100111 (bin) / 39 (dec)	39 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1501	0111001 (bin) / 57 (dec)	0000000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0111010 (bin) / 58 (dec)	58 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
875	0101100 (bin) / 44 (dec)	0111000 (bin) / 56 (dec)	1 (bin) / 1 (dec)	1100101 (bin) / 101 (dec)	101 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1040	1101000 (bin) / 104 (dec)	1011010 (bin) / 90 (dec)	1 (bin) / 1 (dec)	1000011 (bin) / 67 (dec)	67 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
617	0010110 (bin) / 22 (dec)	1110110 (bin) / 118 (dec)	1 (bin) / 1 (dec)	0001101 (bin) / 13 (dec)	13 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
73	0111011 (bin) / 59 (dec)	0001010 (bin) / 10 (dec)	0 (bin) / 0 (dec)	1000101 (bin) / 69 (dec)	69 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
344	1001011 (bin) / 75 (dec)	1011100 (bin) / 92 (dec)	1 (bin) / 1 (dec)	0101000 (bin) / 40 (dec)	40 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1115	0010101 (bin) / 21 (dec)	1100101 (bin) / 101 (dec)	1 (bin) / 1 (dec)	1111011 (bin) / 123 (dec)	123 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
625	0111011 (bin) / 59 (dec)	0011100 (bin) / 28 (dec)	0 (bin) / 0 (dec)	1010111 (bin) / 87 (dec)	87 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1624	1010000 (bin) / 80 (dec)	0110111 (bin) / 55 (dec)	1 (bin) / 1 (dec)	0001000 (bin) / 8 (dec)	8 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
6	0010100 (bin) / 20 (dec)	0011011 (bin) / 27 (dec)	1 (bin) / 1 (dec)	0110000 (bin) / 48 (dec)	48 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1413	1100101 (bin) / 101 (dec)	1111011 (bin) / 123 (dec)	1 (bin) / 1 (dec)	1100001 (bin) / 97 (dec)	97 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
21	0010110 (bin) / 22 (dec)	0110001 (bin) / 49 (dec)	0 (bin) / 0 (dec)	1000111 (bin) / 71 (dec)	71 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
999	1111000 (bin) / 120 (dec)	1001110 (bin) / 78 (dec)	1 (bin) / 1 (dec)	1000111 (bin) / 71 (dec)	71 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
331	1100101 (bin) / 101 (dec)	1000010 (bin) / 66 (dec)	1 (bin) / 1 (dec)	0101000 (bin) / 40 (dec)	40 (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) N8

Total tests: 1639

Passed tests: 1639

Failed tests: 0

Test Case	Input a	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Status
1322	00101110 (bin) / 46 (dec)	00111101 (bin) / 61 (dec)	1 (bin) / 1 (dec)	01101100 (bin) / 108 (dec)	108 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1285	11100001 (bin) / 225 (dec)	11100101 (bin) / 229 (dec)	1 (bin) / 1 (dec)	11000111 (bin) / 199 (dec)	199 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1539	11111011 (bin) / 251 (dec)	10001011 (bin) / 139 (dec)	0 (bin) / 0 (dec)	10000110 (bin) / 134 (dec)	134 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1313	01100100 (bin) / 100 (dec)	00110111 (bin) / 55 (dec)	0 (bin) / 0 (dec)	10011011 (bin) / 155 (dec)	155 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
29	10000111 (bin) / 135 (dec)	00011010 (bin) / 26 (dec)	1 (bin) / 1 (dec)	10100010 (bin) / 162 (dec)	162 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
939	10010100 (bin) / 148 (dec)	00111111 (bin) / 63 (dec)	1 (bin) / 1 (dec)	11010100 (bin) / 212 (dec)	212 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
122	00011011 (bin) / 27 (dec)	01000110 (bin) / 70 (dec)	0 (bin) / 0 (dec)	01100001 (bin) / 97 (dec)	97 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1047	01001000 (bin) / 72 (dec)	10111101 (bin) / 189 (dec)	1 (bin) / 1 (dec)	00000110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1619	10101101 (bin) / 173 (dec)	11010111 (bin) / 215 (dec)	1 (bin) / 1 (dec)	10000101 (bin) / 133 (dec)	133 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
261	10001111 (bin) / 143 (dec)	11101111 (bin) / 239 (dec)	0 (bin) / 0 (dec)	01111110 (bin) / 126 (dec)	126 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
767	10101100 (bin) / 172 (dec)	01011011 (bin) / 91 (dec)	1 (bin) / 1 (dec)	00001000 (bin) / 8 (dec)	8 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
596	01010010 (bin) / 82 (dec)	10010001 (bin) / 145 (dec)	0 (bin) / 0 (dec)	11100011 (bin) / 227 (dec)	227 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
921	11110001 (bin) / 241 (dec)	00100000 (bin) / 32 (dec)	0 (bin) / 0 (dec)	00010001 (bin) / 17 (dec)	17 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
606	11010010 (bin) / 210 (dec)	10010001 (bin) / 145 (dec)	1 (bin) / 1 (dec)	01100100 (bin) / 100 (dec)	100 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
570	01011111 (bin) / 95 (dec)	10000011 (bin) / 131 (dec)	1 (bin) / 1 (dec)	11100011 (bin) / 227 (dec)	227 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
254	01010110 (bin) / 86 (dec)	10001100 (bin) / 140 (dec)	0 (bin) / 0 (dec)	11100010 (bin) / 226 (dec)	226 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
116	10001011 (bin) / 139 (dec)	10010000 (bin) / 144 (dec)	1 (bin) / 1 (dec)	00011100 (bin) / 28 (dec)	28 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
653	10010010 (bin) / 146 (dec)	10011101 (bin) / 157 (dec)	0 (bin) / 0 (dec)	00101111 (bin) / 47 (dec)	47 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
314	01010010 (bin) / 82 (dec)	00000011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	01010101 (bin) / 85 (dec)	85 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1224	01011010 (bin) / 90 (dec)	10111011 (bin) / 187 (dec)	1 (bin) / 1 (dec)	00010110 (bin) / 22 (dec)	22 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1161	11101111 (bin) / 239 (dec)	01111100 (bin) / 124 (dec)	1 (bin) / 1 (dec)	01101100 (bin) / 108 (dec)	108 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1447	10110001 (bin) / 177 (dec)	01110010 (bin) / 114 (dec)	0 (bin) / 0 (dec)	00100011 (bin) / 35 (dec)	35 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
362	00111000 (bin) / 56 (dec)	00011101 (bin) / 29 (dec)	1 (bin) / 1 (dec)	01010110 (bin) / 86 (dec)	86 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
440	10111011 (bin) / 187 (dec)	11000011 (bin) / 195 (dec)	0 (bin) / 0 (dec)	01111110 (bin) / 126 (dec)	126 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1572	01110100 (bin) / 116 (dec)	01011011 (bin) / 91 (dec)	0 (bin) / 0 (dec)	11001111 (bin) / 207 (dec)	207 (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
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
2	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
6	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
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
5	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
0	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
4	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
1	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 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
1	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
0	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
2	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (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
```