Testbenching Report for carry_skip_block

Table of Contents

Testbench for carry_skip_block with parameter(s) WIDTH1 3

Testbench for carry_skip_block with parameter(s) WIDTH2 5

Testbench for carry_skip_block with parameter(s) WIDTH3 10

Testbench for carry_skip_block with parameter(s) WIDTH4 15

Testbench for carry_skip_block with parameter(s) WIDTH5 20

Testbench for carry_skip_block with parameter(s) WIDTH6 25

Testbench for carry_skip_block with parameter(s) WIDTH7 30

Testbench for carry_skip_block with parameter(s) WIDTH8 35

Testbench for full_adder with parameter(s) 33

Testbench for half_adder with parameter(s) 34

Testbench for carry_skip_block with parameter(s) WIDTH1

Total tests: 8

Passed tests: 8

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expected blo
dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (d
dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (d
dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (d
dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (d
dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (d
dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (d
dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (d
dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (d

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
            def generate_expected(self, test_case):
                width = self.bit_width
                carry = [0] * (width + 1)
                carry[0] = test_case["cin"]
                sum_ = [0] * width
                g = [0] * width
                p = [0] * width

                a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
                b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

                for i in range(width):
                    sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
                    carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
                    g[i] = a_bits[i] & b_bits[i]
                    p[i] = a_bits[i] | b_bits[i]

                block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
                cout = carry[width]

                sum_int = sum(sum_[i] << i for i in range(width))

                return {
                    "sum": sum_int,
                    "cout": cout,
                    "block_carry_out": block_carry_out
                }
```

Testbench for carry_skip_block with parameter(s) WIDTH2

Total tests: 32

Passed tests: 32

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expected bl
dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	10 (bin) / 2 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	01 (bin) / 1 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	01 (bin) / 1 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	10 (bin) / 2 (dec)	1 (bin) / 1 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	10 (bin) / 2 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	10 (bin) / 2 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	00 (bin) / 0 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expected bl
dec)	00 (bin) / 0 (dec)	1 (bin) / 1 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	11 (bin) / 3 (dec)	1 (bin) / 1 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	01 (bin) / 1 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	01 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	01 (bin) / 1 (dec)	1 (bin) / 1 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	01 (bin) / 1 (dec)	0 (bin) / 0 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	00 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	11 (bin) / 3 (dec)	0 (bin) / 0 (dec)	11 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	10 (bin) / 2 (dec)	0 (bin) / 0 (dec)	10 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
def generate_expected(self, test_case):
    width = self.bit_width
    carry = [0] * (width + 1)
    carry[0] = test_case["cin"]
    sum_ = [0] * width
    g = [0] * width
    p = [0] * width

    a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
    b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

    for i in range(width):
        sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
        carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
        g[i] = a_bits[i] & b_bits[i]
        p[i] = a_bits[i] | b_bits[i]

    block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
    cout = carry[width]

    sum_int = sum(sum_[i] << i for i in range(width))

    return {
        "sum": sum_int,
        "cout": cout,
        "block_carry_out": block_carry_out
    }
```

Testbench for carry_skip_block with parameter(s) WIDTH3

Total tests: 35

Passed tests: 35

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expected b
dec)	011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	010 (bin) / 2 (dec)	2 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	000 (bin) / 0 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	001 (bin) / 1 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	110 (bin) / 6 (dec)	0 (bin) / 0 (dec)	011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	011 (bin) / 3 (dec)	3 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expected b
dec)	011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0
dec)	111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1
dec)	111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
def generate_expected(self, test_case):
    width = self.bit_width
    carry = [0] * (width + 1)
    carry[0] = test_case["cin"]
    sum_ = [0] * width
    g = [0] * width
    p = [0] * width

    a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
    b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

    for i in range(width):
        sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
        carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
        g[i] = a_bits[i] & b_bits[i]
        p[i] = a_bits[i] | b_bits[i]

    block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
    cout = carry[width]

    sum_int = sum(sum_[i] << i for i in range(width))

    return {
        "sum": sum_int,
        "cout": cout,
        "block_carry_out": block_carry_out
    }
```

Testbench for carry_skip_block with parameter(s) WIDTH4

Total tests: 35

Passed tests: 35

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expecte
lec)	0001 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	0011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	0011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	1000 (bin) / 8 (dec)	8 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1011 (bin) / 11 (dec)	0 (bin) / 0 (dec)	0000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	0010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	1010 (bin) / 10 (dec)	10 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	0101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	1100 (bin) / 12 (dec)	12 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1111 (bin) / 15 (dec)	0 (bin) / 0 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	0010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	0001 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	0101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	0101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	0000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	1111 (bin) / 15 (dec)	1 (bin) / 1 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	1010 (bin) / 10 (dec)	0 (bin) / 0 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1001 (bin) / 9 (dec)	1 (bin) / 1 (dec)	1101 (bin) / 13 (dec)	13 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	1000 (bin) / 8 (dec)	0 (bin) / 0 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	0010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	1100 (bin) / 12 (dec)	12 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1110 (bin) / 14 (dec)	0 (bin) / 0 (dec)	0101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	1110 (bin) / 14 (dec)	0 (bin) / 0 (dec)	0011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	1011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	1010 (bin) / 10 (dec)	10 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	1011 (bin) / 11 (dec)	0 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1010 (bin) / 10 (dec)	1 (bin) / 1 (dec)	1100 (bin) / 12 (dec)	12 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	0101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	1001 (bin) / 9 (dec)	9 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	0010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	1010 (bin) / 10 (dec)	10 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1110 (bin) / 14 (dec)	1 (bin) / 1 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expecte
dec)	0000 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	0010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	0100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	1011 (bin) / 11 (dec)	0 (bin) / 0 (dec)	0011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	0010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	1000 (bin) / 8 (dec)	8 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	1100 (bin) / 12 (dec)	1 (bin) / 1 (dec)	1011 (bin) / 11 (dec)	11 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	1100 (bin) / 12 (dec)	0 (bin) / 0 (dec)	1111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	0110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	1001 (bin) / 9 (dec)	9 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	1001 (bin) / 9 (dec)	0 (bin) / 0 (dec)	0100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	0101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	0110 (bin) / 6 (dec)	6 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	1011 (bin) / 11 (dec)	1 (bin) / 1 (dec)	0111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	0111 (bin) / 7 (dec)	1 (bin) / 1 (dec)	1011 (bin) / 11 (dec)	11 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
def generate_expected(self, test_case):
    width = self.bit_width
    carry = [0] * (width + 1)
    carry[0] = test_case["cin"]
    sum_ = [0] * width
    g = [0] * width
    p = [0] * width

    a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
    b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

    for i in range(width):
        sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
        carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
        g[i] = a_bits[i] & b_bits[i]
        p[i] = a_bits[i] | b_bits[i]

    block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
    cout = carry[width]

    sum_int = sum(sum_[i] << i for i in range(width))

    return {
        "sum": sum_int,
        "cout": cout,
        "block_carry_out": block_carry_out
    }
```

Testbench for carry_skip_block with parameter(s) WIDTH5

Total tests: 35

Passed tests: 35

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expect
dec)	10101 (bin) / 21 (dec)	0 (bin) / 0 (dec)	00111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	10111 (bin) / 23 (dec)	0 (bin) / 0 (dec)	10100 (bin) / 20 (dec)	20 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	00010 (bin) / 2 (dec)	0 (bin) / 0 (dec)	11100 (bin) / 28 (dec)	28 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	00111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	11111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	11101 (bin) / 29 (dec)	1 (bin) / 1 (dec)	00100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	10011 (bin) / 19 (dec)	0 (bin) / 0 (dec)	10010 (bin) / 18 (dec)	18 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	11000 (bin) / 24 (dec)	1 (bin) / 1 (dec)	10110 (bin) / 22 (dec)	22 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	00111 (bin) / 7 (dec)	0 (bin) / 0 (dec)	01111 (bin) / 15 (dec)	15 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	10011 (bin) / 19 (dec)	0 (bin) / 0 (dec)	10101 (bin) / 21 (dec)	21 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	01110 (bin) / 14 (dec)	0 (bin) / 0 (dec)	11111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	11110 (bin) / 30 (dec)	0 (bin) / 0 (dec)	00100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	00011 (bin) / 3 (dec)	0 (bin) / 0 (dec)	00101 (bin) / 5 (dec)	5 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	00100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	10011 (bin) / 19 (dec)	19 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	01110 (bin) / 14 (dec)	0 (bin) / 0 (dec)	01011 (bin) / 11 (dec)	11 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	01010 (bin) / 10 (dec)	0 (bin) / 0 (dec)	11111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	10001 (bin) / 17 (dec)	1 (bin) / 1 (dec)	10101 (bin) / 21 (dec)	21 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	00011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	01101 (bin) / 13 (dec)	13 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	00101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	11101 (bin) / 29 (dec)	29 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	10010 (bin) / 18 (dec)	0 (bin) / 0 (dec)	00101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	10101 (bin) / 21 (dec)	1 (bin) / 1 (dec)	11100 (bin) / 28 (dec)	28 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	01000 (bin) / 8 (dec)	1 (bin) / 1 (dec)	01100 (bin) / 12 (dec)	12 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	00100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	11101 (bin) / 29 (dec)	29 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	10000 (bin) / 16 (dec)	1 (bin) / 1 (dec)	11010 (bin) / 26 (dec)	26 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expect
dec)	11001 (bin) / 25 (dec)	0 (bin) / 0 (dec)	01110 (bin) / 14 (dec)	14 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	10000 (bin) / 16 (dec)	0 (bin) / 0 (dec)	10010 (bin) / 18 (dec)	18 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	00101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	01001 (bin) / 9 (dec)	9 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	00100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	11100 (bin) / 28 (dec)	28 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	11001 (bin) / 25 (dec)	1 (bin) / 1 (dec)	00110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	11100 (bin) / 28 (dec)	0 (bin) / 0 (dec)	10000 (bin) / 16 (dec)	16 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	11000 (bin) / 24 (dec)	1 (bin) / 1 (dec)	00100 (bin) / 4 (dec)	4 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	01010 (bin) / 10 (dec)	0 (bin) / 0 (dec)	11000 (bin) / 24 (dec)	24 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	01101 (bin) / 13 (dec)	0 (bin) / 0 (dec)	10101 (bin) / 21 (dec)	21 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
lec)	11100 (bin) / 28 (dec)	0 (bin) / 0 (dec)	00000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	11000 (bin) / 24 (dec)	0 (bin) / 0 (dec)	01100 (bin) / 12 (dec)	12 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
lec)	01110 (bin) / 14 (dec)	1 (bin) / 1 (dec)	10100 (bin) / 20 (dec)	20 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
def generate_expected(self, test_case):
    width = self.bit_width
    carry = [0] * (width + 1)
    carry[0] = test_case["cin"]
    sum_ = [0] * width
    g = [0] * width
    p = [0] * width

    a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
    b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

    for i in range(width):
        sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
        carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
        g[i] = a_bits[i] & b_bits[i]
        p[i] = a_bits[i] | b_bits[i]

    block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
    cout = carry[width]

    sum_int = sum(sum_[i] << i for i in range(width))

    return {
        "sum": sum_int,
        "cout": cout,
        "block_carry_out": block_carry_out
    }
```

Testbench for carry_skip_block with parameter(s) WIDTH6

Total tests: 35

Passed tests: 35

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expe
dec)	001110 (bin) / 14 (dec)	0 (bin) / 0 (dec)	110001 (bin) / 49 (dec)	49 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	000110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	000010 (bin) / 2 (dec)	2 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	110010 (bin) / 50 (dec)	0 (bin) / 0 (dec)	001001 (bin) / 9 (dec)	9 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	101011 (bin) / 43 (dec)	0 (bin) / 0 (dec)	001110 (bin) / 14 (dec)	14 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	100001 (bin) / 33 (dec)	1 (bin) / 1 (dec)	000111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	010010 (bin) / 18 (dec)	0 (bin) / 0 (dec)	101100 (bin) / 44 (dec)	44 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	011011 (bin) / 27 (dec)	0 (bin) / 0 (dec)	101000 (bin) / 40 (dec)	40 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	111100 (bin) / 60 (dec)	0 (bin) / 0 (dec)	000110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	001011 (bin) / 11 (dec)	0 (bin) / 0 (dec)	010111 (bin) / 23 (dec)	23 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	110001 (bin) / 49 (dec)	1 (bin) / 1 (dec)	011000 (bin) / 24 (dec)	24 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	111111 (bin) / 63 (dec)	0 (bin) / 0 (dec)	110100 (bin) / 52 (dec)	52 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	001111 (bin) / 15 (dec)	1 (bin) / 1 (dec)	011111 (bin) / 31 (dec)	31 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	010111 (bin) / 23 (dec)	1 (bin) / 1 (dec)	111100 (bin) / 60 (dec)	60 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	011010 (bin) / 26 (dec)	0 (bin) / 0 (dec)	100111 (bin) / 39 (dec)	39 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	110100 (bin) / 52 (dec)	1 (bin) / 1 (dec)	100001 (bin) / 33 (dec)	33 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	101001 (bin) / 41 (dec)	1 (bin) / 1 (dec)	100101 (bin) / 37 (dec)	37 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
ec)	011100 (bin) / 28 (dec)	1 (bin) / 1 (dec)	011110 (bin) / 30 (dec)	30 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	100011 (bin) / 35 (dec)	0 (bin) / 0 (dec)	111100 (bin) / 60 (dec)	60 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	101010 (bin) / 42 (dec)	0 (bin) / 0 (dec)	001101 (bin) / 13 (dec)	13 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	010111 (bin) / 23 (dec)	1 (bin) / 1 (dec)	001111 (bin) / 15 (dec)	15 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	010100 (bin) / 20 (dec)	0 (bin) / 0 (dec)	111100 (bin) / 60 (dec)	60 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	001100 (bin) / 12 (dec)	1 (bin) / 1 (dec)	000000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	111010 (bin) / 58 (dec)	0 (bin) / 0 (dec)	010000 (bin) / 16 (dec)	16 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Expe
dec)	101110 (bin) / 46 (dec)	1 (bin) / 1 (dec)	111111 (bin) / 63 (dec)	63 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	101001 (bin) / 41 (dec)	1 (bin) / 1 (dec)	101000 (bin) / 40 (dec)	40 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	111011 (bin) / 59 (dec)	0 (bin) / 0 (dec)	110010 (bin) / 50 (dec)	50 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	000110 (bin) / 6 (dec)	1 (bin) / 1 (dec)	010110 (bin) / 22 (dec)	22 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
dec)	101100 (bin) / 44 (dec)	1 (bin) / 1 (dec)	000000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	111001 (bin) / 57 (dec)	0 (bin) / 0 (dec)	000110 (bin) / 6 (dec)	6 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	101010 (bin) / 42 (dec)	1 (bin) / 1 (dec)	000011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	110000 (bin) / 48 (dec)	1 (bin) / 1 (dec)	010111 (bin) / 23 (dec)	23 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	101100 (bin) / 44 (dec)	1 (bin) / 1 (dec)	100000 (bin) / 32 (dec)	32 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	100010 (bin) / 34 (dec)	0 (bin) / 0 (dec)	011100 (bin) / 28 (dec)	28 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
dec)	001111 (bin) / 15 (dec)	0 (bin) / 0 (dec)	011101 (bin) / 29 (dec)	29 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
ec)	101001 (bin) / 41 (dec)	0 (bin) / 0 (dec)	101100 (bin) / 44 (dec)	44 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
def generate_expected(self, test_case):
    width = self.bit_width
    carry = [0] * (width + 1)
    carry[0] = test_case["cin"]
    sum_ = [0] * width
    g = [0] * width
    p = [0] * width

    a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
    b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

    for i in range(width):
        sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
        carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
        g[i] = a_bits[i] & b_bits[i]
        p[i] = a_bits[i] | b_bits[i]

    block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
    cout = carry[width]

    sum_int = sum(sum_[i] << i for i in range(width))

    return {
        "sum": sum_int,
        "cout": cout,
        "block_carry_out": block_carry_out
    }
```

Testbench for carry_skip_block with parameter(s) WIDTH7

Total tests: 35

Passed tests: 35

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	E
c)	1001111 (bin) / 79 (dec)	1 (bin) / 1 (dec)	1011011 (bin) / 91 (dec)	91 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0110000 (bin) / 48 (dec)	0 (bin) / 0 (dec)	1110011 (bin) / 115 (dec)	115 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0000011 (bin) / 3 (dec)	1 (bin) / 1 (dec)	0000100 (bin) / 4 (dec)	4 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	1011110 (bin) / 94 (dec)	1 (bin) / 1 (dec)	1101000 (bin) / 104 (dec)	104 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	1101111 (bin) / 111 (dec)	1 (bin) / 1 (dec)	0111001 (bin) / 57 (dec)	57 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
ec)	0001001 (bin) / 9 (dec)	1 (bin) / 1 (dec)	0000101 (bin) / 5 (dec)	5 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	0100111 (bin) / 39 (dec)	1 (bin) / 1 (dec)	1110100 (bin) / 116 (dec)	116 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	1000001 (bin) / 65 (dec)	0 (bin) / 0 (dec)	1000001 (bin) / 65 (dec)	65 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0101011 (bin) / 43 (dec)	0 (bin) / 0 (dec)	1101110 (bin) / 110 (dec)	110 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0000010 (bin) / 2 (dec)	1 (bin) / 1 (dec)	0011101 (bin) / 29 (dec)	29 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0011000 (bin) / 24 (dec)	1 (bin) / 1 (dec)	0011110 (bin) / 30 (dec)	30 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0000101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	1000011 (bin) / 67 (dec)	67 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
ec)	0100111 (bin) / 39 (dec)	1 (bin) / 1 (dec)	0010011 (bin) / 19 (dec)	19 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	1111010 (bin) / 122 (dec)	0 (bin) / 0 (dec)	0010100 (bin) / 20 (dec)	20 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	1100011 (bin) / 99 (dec)	0 (bin) / 0 (dec)	0001101 (bin) / 13 (dec)	13 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	1111111 (bin) / 127 (dec)	0 (bin) / 0 (dec)	1011100 (bin) / 92 (dec)	92 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	0000100 (bin) / 4 (dec)	1 (bin) / 1 (dec)	1100111 (bin) / 103 (dec)	103 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0100001 (bin) / 33 (dec)	0 (bin) / 0 (dec)	1110110 (bin) / 118 (dec)	118 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
ec)	0000101 (bin) / 5 (dec)	0 (bin) / 0 (dec)	0000011 (bin) / 3 (dec)	3 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	0111000 (bin) / 56 (dec)	1 (bin) / 1 (dec)	1100101 (bin) / 101 (dec)	101 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	1101000 (bin) / 104 (dec)	0 (bin) / 0 (dec)	1101010 (bin) / 106 (dec)	106 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	1001111 (bin) / 79 (dec)	1 (bin) / 1 (dec)	1110111 (bin) / 119 (dec)	119 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0110011 (bin) / 51 (dec)	1 (bin) / 1 (dec)	1101111 (bin) / 111 (dec)	111 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	Ex
ec)	1101111 (bin) / 111 (dec)	1 (bin) / 1 (dec)	1101001 (bin) / 105 (dec)	105 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	1100111 (bin) / 103 (dec)	0 (bin) / 0 (dec)	1111110 (bin) / 126 (dec)	126 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	1101110 (bin) / 110 (dec)	1 (bin) / 1 (dec)	0110010 (bin) / 50 (dec)	50 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	1011111 (bin) / 95 (dec)	1 (bin) / 1 (dec)	0010011 (bin) / 19 (dec)	19 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	1010010 (bin) / 82 (dec)	1 (bin) / 1 (dec)	0100010 (bin) / 34 (dec)	34 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	0001111 (bin) / 15 (dec)	0 (bin) / 0 (dec)	1011100 (bin) / 92 (dec)	92 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0000001 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1010111 (bin) / 87 (dec)	87 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	0110101 (bin) / 53 (dec)	0 (bin) / 0 (dec)	0111111 (bin) / 63 (dec)	63 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	1001111 (bin) / 79 (dec)	1 (bin) / 1 (dec)	0101001 (bin) / 41 (dec)	41 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	1010011 (bin) / 83 (dec)	1 (bin) / 1 (dec)	0011000 (bin) / 24 (dec)	24 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	0010010 (bin) / 18 (dec)	1 (bin) / 1 (dec)	1100010 (bin) / 98 (dec)	98 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
ec)	1010101 (bin) / 85 (dec)	0 (bin) / 0 (dec)	1000000 (bin) / 64 (dec)	64 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
def generate_expected(self, test_case):
    width = self.bit_width
    carry = [0] * (width + 1)
    carry[0] = test_case["cin"]
    sum_ = [0] * width
    g = [0] * width
    p = [0] * width

    a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
    b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

    for i in range(width):
        sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
        carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
        g[i] = a_bits[i] & b_bits[i]
        p[i] = a_bits[i] | b_bits[i]

    block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
    cout = carry[width]

    sum_int = sum(sum_[i] << i for i in range(width))

    return {
        "sum": sum_int,
        "cout": cout,
        "block_carry_out": block_carry_out
    }
```

Testbench for carry_skip_block with parameter(s) WIDTH8

Total tests: 35

Passed tests: 35

Failed tests: 0

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	
c)	10110110 (bin) / 182 (dec)	1 (bin) / 1 (dec)	11110000 (bin) / 240 (dec)	240 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	00111010 (bin) / 58 (dec)	1 (bin) / 1 (dec)	00101101 (bin) / 45 (dec)	45 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	11111111 (bin) / 255 (dec)	1 (bin) / 1 (dec)	10011110 (bin) / 158 (dec)	158 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	01110110 (bin) / 118 (dec)	1 (bin) / 1 (dec)	11000101 (bin) / 197 (dec)	197 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	10110110 (bin) / 182 (dec)	1 (bin) / 1 (dec)	01110010 (bin) / 114 (dec)	114 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	00001101 (bin) / 13 (dec)	0 (bin) / 0 (dec)	01010110 (bin) / 86 (dec)	86 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	00001001 (bin) / 9 (dec)	1 (bin) / 1 (dec)	11101100 (bin) / 236 (dec)	236 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	11010110 (bin) / 214 (dec)	0 (bin) / 0 (dec)	00101110 (bin) / 46 (dec)	46 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
)	01010111 (bin) / 87 (dec)	0 (bin) / 0 (dec)	01011100 (bin) / 92 (dec)	92 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	01011111 (bin) / 95 (dec)	1 (bin) / 1 (dec)	00011010 (bin) / 26 (dec)	26 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	10111111 (bin) / 191 (dec)	0 (bin) / 0 (dec)	00000111 (bin) / 7 (dec)	7 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	11110110 (bin) / 246 (dec)	0 (bin) / 0 (dec)	11110100 (bin) / 244 (dec)	244 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	10111000 (bin) / 184 (dec)	1 (bin) / 1 (dec)	01110110 (bin) / 118 (dec)	118 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	10111010 (bin) / 186 (dec)	0 (bin) / 0 (dec)	10001010 (bin) / 138 (dec)	138 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	11010010 (bin) / 210 (dec)	1 (bin) / 1 (dec)	00111110 (bin) / 62 (dec)	62 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	10010110 (bin) / 150 (dec)	0 (bin) / 0 (dec)	00101001 (bin) / 41 (dec)	41 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
)	00010110 (bin) / 22 (dec)	1 (bin) / 1 (dec)	00011101 (bin) / 29 (dec)	29 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	01111110 (bin) / 126 (dec)	0 (bin) / 0 (dec)	10010011 (bin) / 147 (dec)	147 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	00100001 (bin) / 33 (dec)	1 (bin) / 1 (dec)	00111111 (bin) / 63 (dec)	63 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	11011111 (bin) / 223 (dec)	1 (bin) / 1 (dec)	01000001 (bin) / 65 (dec)	65 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	11110000 (bin) / 240 (dec)	0 (bin) / 0 (dec)	00011001 (bin) / 25 (dec)	25 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	01111111 (bin) / 127 (dec)	0 (bin) / 0 (dec)	10101000 (bin) / 168 (dec)	168 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	00000100 (bin) / 4 (dec)	0 (bin) / 0 (dec)	10000101 (bin) / 133 (dec)	133 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	

	Input b	Input cin	Output sum (Actual)	Expected sum	Output cout (Actual)	Expected cout	Output block_carry_out (Actual)	
c)	01010101 (bin) / 85 (dec)	1 (bin) / 1 (dec)	01110111 (bin) / 119 (dec)	119 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	01111001 (bin) / 121 (dec)	1 (bin) / 1 (dec)	11000110 (bin) / 198 (dec)	198 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	01000100 (bin) / 68 (dec)	0 (bin) / 0 (dec)	10010100 (bin) / 148 (dec)	148 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	01011000 (bin) / 88 (dec)	1 (bin) / 1 (dec)	00110100 (bin) / 52 (dec)	52 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	01000101 (bin) / 69 (dec)	1 (bin) / 1 (dec)	01000101 (bin) / 69 (dec)	69 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	00000101 (bin) / 5 (dec)	1 (bin) / 1 (dec)	10011110 (bin) / 158 (dec)	158 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	
c)	11001101 (bin) / 205 (dec)	1 (bin) / 1 (dec)	01010010 (bin) / 82 (dec)	82 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	11001110 (bin) / 206 (dec)	1 (bin) / 1 (dec)	00101100 (bin) / 44 (dec)	44 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	11111100 (bin) / 252 (dec)	1 (bin) / 1 (dec)	10001110 (bin) / 142 (dec)	142 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	10110000 (bin) / 176 (dec)	1 (bin) / 1 (dec)	00000000 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	10001110 (bin) / 142 (dec)	0 (bin) / 0 (dec)	00101110 (bin) / 46 (dec)	46 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	
c)	10110010 (bin) / 178 (dec)	1 (bin) / 1 (dec)	01011001 (bin) / 89 (dec)	89 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	

Rule: CarrySkipBlockRule

Input Variables: a, b, cin

Output Variables: sum, cout, block_carry_out

Bit Width: 4

Pattern: StringMatchPattern

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

Generate expected values function:

```
def generate_expected(self, test_case):
    width = self.bit_width
    carry = [0] * (width + 1)
    carry[0] = test_case["cin"]
    sum_ = [0] * width
    g = [0] * width
    p = [0] * width

    a_bits = [(test_case["a"] >> i) & 1 for i in range(width)]
    b_bits = [(test_case["b"] >> i) & 1 for i in range(width)]

    for i in range(width):
        sum_[i] = (a_bits[i] + b_bits[i] + carry[i]) & 1
        carry[i + 1] = (a_bits[i] + b_bits[i] + carry[i]) >> 1
        g[i] = a_bits[i] & b_bits[i]
        p[i] = a_bits[i] | b_bits[i]

    block_carry_out = g[width-1] | (p[width-1] & carry[width-1])
    cout = carry[width]

    sum_int = sum(sum_[i] << i for i in range(width))

    return {
        "sum": sum_int,
        "cout": cout,
        "block_carry_out": block_carry_out
    }
```

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
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
1	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
2	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
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
4	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
6	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
7	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

Testbench for half_adder with parameter(s)

Total tests: 4

Passed tests: 4

Failed tests: 0

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