Testbenching Report for kogge_stone_adder

Table of Contents

Testbench Summary

Component	Total Tests	Passed	Failed
kogge_stone_adder_N8	0	0	0
kogge_stone_adder_N16	0	0	0
kogge_stone_adder_N32	0	0	0
kogge_stone_adder_N64	0	0	0
big_circle_	16	16	0
small_circle_	2	2	0
square_	4	4	0
triangle_	4	4	0

Testbench for kogge_stone_adder with parameter(s) N8

Total tests: 0

Passed tests: 0

Failed tests: 0

Test Case	Status

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 kogge_stone_adder with parameter(s) N16

Total tests: 0

Passed tests: 0

Failed tests: 0

Test Case	Status

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 kogge_stone_adder with parameter(s) N32

Total tests: 0

Passed tests: 0

Failed tests: 0

Test Case	Status

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 kogge_stone_adder with parameter(s) N64

Total tests: 0

Passed tests: 0

Failed tests: 0

Test Case	Status

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 big_circle with parameter(s)

Total tests: 16

Passed tests: 16

Failed tests: 0

Test Case	Input Gi	Input Pi	Input GiPrev	Input PiPrev	Output G (Actual)	Expected G	Output P (Actual)	Expected P	Status
0	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	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)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
4	1 (bin) / 1 (dec)	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
5	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
6	1 (bin) / 1 (dec)	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	0 (bin) / 0 (dec)	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
8	0 (bin) / 0 (dec)	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
9	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
10	1 (bin) / 1 (dec)	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
11	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
12	1 (bin) / 1 (dec)	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
13	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
14	0 (bin) / 0 (dec)	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
15	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: BigCircleRule

Input Variables: Gi, Pi, GiPrev, PiPrev

Output Variables: G, P

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
                G = [0] * width
                P = [0] * width

                for i in range(width):
                    e = (test_case["Pi"] >> i) & 1 & (test_case["GiPrev"] >> i) & 1
                    G[i] = e | (test_case["Gi"] >> i) & 1
                    P[i] = (test_case["Pi"] >> i) & 1 & (test_case["PiPrev"] >> i) & 1

                G_int = sum(G[i] << i for i in range(width))
                P_int = sum(P[i] << i for i in range(width))

                return {
                    "G": G_int,
                    "P": P_int
                }
```

Testbench for small_circle with parameter(s)

Total tests: 2

Passed tests: 2

Failed tests: 0

Test Case	Input Gi	Output Ci (Actual)	Expected Ci	Status
0	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: SmallCircleRule

Input Variables: Gi

Output Variables: Ci

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
    Ci = [(test_case["Gi"] >> i) & 1 for i in range(width)]

    Ci_int = sum(Ci[i] << i for i in range(width))

    return {
        "Ci": Ci_int
    }
```

Testbench for square with parameter(s)

Total tests: 4

Passed tests: 4

Failed tests: 0

Test Case	Input Ai	Input Bi	Output G (Actual)	Expected G	Output P (Actual)	Expected P	Status
0	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
1	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
2	1 (bin) / 1 (dec)	1 (bin) / 1 (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)	0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed

Rule: SquareRule

Input Variables: Ai, Bi

Output Variables: G, P

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):
    a_bits = [(test_case["Ai"] >> i) & 1 for i in range(self.bit_width)]
    b_bits = [(test_case["Bi"] >> i) & 1 for i in range(self.bit_width)]

    G = [a_bits[i] & b_bits[i] for i in range(self.bit_width)]
    P = [a_bits[i] ^ b_bits[i] for i in range(self.bit_width)]

    G_int = sum(G[i] << i for i in range(self.bit_width))
    P_int = sum(P[i] << i for i in range(self.bit_width))

    return {
        "G": G_int,
        "P": P_int
    }
```

Testbench for triangle with parameter(s)

Total tests: 4

Passed tests: 4

Failed tests: 0

Test Case	Input Pi	Input CiPrev	Output Si (Actual)	Expected Si	Status
0	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
1	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
2	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed
3	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed

Rule: TriangleRule

Input Variables: Pi, CiPrev

Output Variables: Si

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):
    P_bits = [(test_case["Pi"] >> i) & 1 for i in range(self.bit_width)]
    CiPrev_bits = [(test_case["CiPrev"] >> i) & 1 for i in range(self.bit_width)]

    S = [P_bits[i] ^ CiPrev_bits[i] for i in range(self.bit_width)]

    S_int = sum(S[i] << i for i in range(self.bit_width))

    return {
        "Si": S_int
    }
```