Testbenching Report for full_subtractor

Table of Contents

Testbench Summary

Component	Total Tests	Passed	Failed
full_subtractor_	8	8	0
half_subtractor_	4	4	0

Testbench for full_subtractor with parameter(s)

Total tests: 8

Passed tests: 8

Failed tests: 0

Test Case	Input a	Input b	Input bin	Output diff (Actual)	Expected diff	Output bout (Actual)	Expected bout	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	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)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (dec)	0 (bin) / 0 (dec)	0 (dec)	Passed
3	1 (bin) / 1 (dec)	1 (bin) / 1 (dec)	0 (bin) / 0 (dec)	0 (bin) / 0 (dec)	0 (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)	0 (bin) / 0 (dec)	0 (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
6	0 (bin) / 0 (dec)	0 (bin) / 0 (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)	0 (bin) / 0 (dec)	1 (bin) / 1 (dec)	1 (dec)	1 (bin) / 1 (dec)	1 (dec)	Passed

Rule: SubtractorRule

Input Variables: a, b, bin

Output Variables: diff, bout

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 "bin" in test_case:
                    diff_val = test_case["a"] - test_case["b"] - test_case["bin"]
                    if diff_val < 0:
                        diff_val += (1 << self.bit_width)
                        bout = 1
                    else:
                        bout = 0
                    outs = {
                        "diff": diff_val & max_val,
                        "bout": bout
                    }
                else:
                    diff_val = test_case["a"] - test_case["b"]
                    if diff_val < 0:
                        diff_val += (1 << self.bit_width)
                        bout = 1
                    else:
                        bout = 0
                    outs = {
                        "diff": diff_val & max_val,
                        "bout": bout
                    }
                return outs
```

Testbench for half_subtractor with parameter(s)

Total tests: 4

Passed tests: 4

Failed tests: 0

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

Rule: SubtractorRule

Input Variables: a, b, bin

Output Variables: diff, bout

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 "bin" in test_case:
        diff_val = test_case["a"] - test_case["b"] - test_case["bin"]
        if diff_val < 0:
            diff_val += (1 << self.bit_width)
            bout = 1
        else:
            bout = 0
        outs = {
            "diff": diff_val & max_val,
            "bout": bout
        }
    else:
        diff_val = test_case["a"] - test_case["b"]
        if diff_val < 0:
            diff_val += (1 << self.bit_width)
            bout = 1
        else:
            bout = 0
        outs = {
            "diff": diff_val & max_val,
            "bout": bout
        }
    return outs
```