Nicole Nechita - rone8293

Simon Nilsson - sini3794

Set Bits test (1.1) - set_tests-py

All qubits are initially set to value 0. Each row means a new test. We expect it to work with increasing qubits.

Value insert	Value Gotten	Expected
0	0	0
1	1	1
10	10	10
000	000	000
101	101	101
100	100	100
0001	0001	0001
0101	0101	0101
1010	1010	1010
1010110110	1010110110	1010110110
101001111010101110111101	101001111010101110111101	101001111010101110111101

Copy Function test (1.2) - *copy_experiment_script.py*

Input qubits A values	Actual B Values after copy	Expected B values after copy
0	0	0
1	1	1
01	01	01
011	011	011
0101	0101	0101
0001	0001	0001
0101101	0101101	0101101
010110110100	010110110100	010110110100

Full Adder test (1.3) adder-test.py

Value a	Value b	Value c_in	Gotten/ Expected r	Gotten/ Expected c_out
0	0	0	0/0	0/0
1	0	0	1/1	0/0
0	1	0	1/1	0/0
0	0	1	1/1	0/0
1	1	0	0/0	1/1
1	1	1	1/1	1/1
1	0	1	0/0	1/1
0	1	1	0/0	1/1

Addition Tests (1.4) - Addition_n_Subtraction_tests.py

NOTE : when 1+1 at the most significant bit, carry_out bit is lost to non-existing further bit (ex: 1000+1000 becomes 0000 instead of 10000).

Value A	Value B	Gotten Value	Expected Value
0	0	0	0
1	0	1	1
01	00	01	01
01	01	10	10
0001	0001	0010	0010
0000	0000	0000	0000
0001	1111	0000	0000
1000	1111	0111	0111
1111	1111	1110	1110
0101	0001	0110	0110
10001	00001	10010	10010
11111	11111	11110	11110
00001	11111	00000	00000
00101	00001	00110	00110

Subtraction Tests (1.5) Addition_n_Subtraction_tests.py

A Value	B Value	Gotten Value	Expected Value
1	1	0	0
01	01	00	00
10	01	01	01
010	001	001	001
0001	0001	0000	0000
0011	0010	0001	0001
1111	1111	0000	0000
0000	0000	0000	0000
1111	0001	1110	1110
1001	0111	0010	0010
0101	0001	0100	0100
1110	1101	0001	0001
10101	00101	10000	10000
11111	11111	00000	00000
01111	00011	01100	01100
10101	10001	00100	00100

Greater than or equal tests (1.6) - Greater_than_test.py

A Value	B Value	Gotten/Expected
0	1	0/0
11	11	1/1
101	110	0/0
1010	1001	1/1
1111	1111	1/1
1111	1110	1/1
1110	1111	0/0
1010	0101	1/1
0101	1011	0/0
0011	0001	1/1
0101	1011	0/0
0011	0001	1/1
0110	0110	1/1
0000	1111	0/0
1111	0000	1/1
10100	10010	1/1

Add Mod Tests (1.7) - add_mod_tests.py

NOTE: Because of the nature of addition stated above where ex:1000+1000 = 0000, instances where this happens leads to strange results after using modulus on them.

A Value	B Value	Mod(N) Value	Gotten	Expected
00	00	00	00	00
10	01	11	00	00
01	01	11	10	10
01	00	11	01	01
000	000	001	000	000
101	010	111	000	000
010	010	110	100	100
010	011	111	101	101
0000	0000	0001	0000	0000
0001	0001	0010	0000	0000
0001	0001	0011	0010	0010
0011	0011	0101	0001	0001
0110	0011	0111	0010	0010
0000	0001	0010	0001	0001
0001	0101	1010	0110	0110
0110	0111	1011	0010	0010
0000	0001	0001	0000	0000
0100	0101	0111	0010	0010
0110	0111	1111	1101	1101

double_mod Tests (1.8) - double_mod_tests.py

NOTE: Because of the nature of addition stated above where ex:1000+1000 = 0000, instances where this happens leads to strange results after using modulus on them.

A Value	N Value	Gotten Value	Expected Value
01	10	00	00
001	010	000	000
011	111	110	110
0000	0001	0000	0000
0001	0010	0000	0000
0001	0011	0010	0010
0010	0011	0001	0001
0001	0101	0010	0010
0100	0101	0011	0011
0100	0111	0001	0001
0101	0111	0011	0011
0110	0111	0101	0101
0101	1011	1010	1010
0101	1010	0000	0000