Truth Table Unit Tests for Four-Bit Adder Project

AND Gate Truth Table Test

The truth table for the AND gate is as follows:  
| A | B | A AND B |  
|---|---|---------|  
| 0 | 0 | 0 |  
| 0 | 1 | 0 |  
| 1 | 0 | 0 |  
| 1 | 1 | 1 |

Test code:

TEST(ANDTruthTableTest, test005) {  
 AND and1(1);  
   
 const unsigned short int bits[2] = {0, 1};  
   
 for (int idx = 0; idx < 2; idx++) {  
 for (int idx2 = 0; idx2 < 2; idx2++) {  
 and1.set\_in1(bits[idx]);  
 and1.set\_in2(bits[idx2]);  
 ASSERT\_EQ(bits[idx] & bits[idx2], and1.out());  
 }  
 }  
}

NAND Gate Truth Table Test

The truth table for the NAND gate is as follows:  
| A | B | A NAND B |  
|---|---|----------|  
| 0 | 0 | 1 |  
| 0 | 1 | 1 |  
| 1 | 0 | 1 |  
| 1 | 1 | 0 |

Test code:

TEST(NANDTruthTableTest, test006) {  
 NAND nand1(1);  
   
 const unsigned short int bits[2] = {0, 1};  
   
 for (int idx = 0; idx < 2; idx++) {  
 for (int idx2 = 0; idx2 < 2; idx2++) {  
 nand1.set\_in1(bits[idx]);  
 nand1.set\_in2(bits[idx2]);  
 ASSERT\_EQ(!(bits[idx] & bits[idx2]), nand1.out());  
 }  
 }  
}

OR Gate Truth Table Test (Already Provided)

The truth table for the OR gate is as follows:  
| A | B | A OR B |  
|---|---|--------|  
| 0 | 0 | 0 |  
| 0 | 1 | 1 |  
| 1 | 0 | 1 |  
| 1 | 1 | 1 |

Test code:

TEST(ORTruthTableTest, test004) {  
 OR or1(1);  
  
 const unsigned short int bits[2] = {0, 1};  
  
 for (int idx = 0; idx < 2; idx++) {  
 for (int idx2 = 0; idx2 < 2; idx2++) {  
 or1.set\_in1(bits[idx]);  
 or1.set\_in2(bits[idx2]);  
 ASSERT\_EQ(bits[idx] | bits[idx2], or1.out());  
 }  
 }  
}

Half-Adder Truth Table Test

The truth table for the half-adder is as follows:  
| A | B | Sum | Carry |  
|---|---|-----|-------|  
| 0 | 0 | 0 | 0 |  
| 0 | 1 | 1 | 0 |  
| 1 | 0 | 1 | 0 |  
| 1 | 1 | 0 | 1 |

Test code:

TEST(HalfAdderTruthTableTest, test007) {  
 HalfAdder ha1;  
   
 const unsigned short int bits[2] = {0, 1};  
   
 for (int idx = 0; idx < 2; idx++) {  
 for (int idx2 = 0; idx2 < 2; idx2++) {  
 ha1.set\_in1(bits[idx]);  
 ha1.set\_in2(bits[idx2]);  
 ASSERT\_EQ((bits[idx] ^ bits[idx2]), ha1.sum()); // XOR for sum  
 ASSERT\_EQ(bits[idx] & bits[idx2], ha1.carry()); // AND for carry  
 }  
 }  
}

Full-Adder Truth Table Test

The truth table for the full-adder is as follows:  
| A | B | C-in | Sum | Carry-out |  
|---|---|------|-----|-----------|  
| 0 | 0 | 0 | 0 | 0 |  
| 0 | 0 | 1 | 1 | 0 |  
| 0 | 1 | 0 | 1 | 0 |  
| 0 | 1 | 1 | 0 | 1 |  
| 1 | 0 | 0 | 1 | 0 |  
| 1 | 0 | 1 | 0 | 1 |  
| 1 | 1 | 0 | 0 | 1 |  
| 1 | 1 | 1 | 1 | 1 |

Test code:

TEST(FullAdderTruthTableTest, test008) {  
 FullAdder fa1;  
   
 const unsigned short int bits[2] = {0, 1};  
   
 for (int idx = 0; idx < 2; idx++) {  
 for (int idx2 = 0; idx2 < 2; idx2++) {  
 for (int cin = 0; cin < 2; cin++) {  
 fa1.set\_in1(bits[idx]);  
 fa1.set\_in2(bits[idx2]);  
 fa1.set\_cin(cin);  
 ASSERT\_EQ((bits[idx] ^ bits[idx2]) ^ cin, fa1.sum()); // XOR for sum  
 ASSERT\_EQ((bits[idx] & bits[idx2]) | ((bits[idx] ^ bits[idx2]) & cin), fa1.carry()); // Carry logic  
 }  
 }  
 }  
}

Four-Bit Adder Truth Table Test

Test code for a four-bit adder:

TEST(FourBitAdderTest, test009) {  
 FourBitAdder fba1;  
   
 unsigned short int A = 0b1010; // Example 4-bit number  
 unsigned short int B = 0b0101; // Example 4-bit number  
   
 fba1.set\_in1(A);  
 fba1.set\_in2(B);  
   
 ASSERT\_EQ(0b1111, fba1.sum()); // Expecting sum of 1010 + 0101 = 1111  
 ASSERT\_EQ(0, fba1.carry()); // No carry for this example  
   
 // You can add more combinations like 0b1111 + 0b0001, etc.  
}