

# Equivalence of Logic Circuits and their Description in VHDL

*Hiroshi Yamazaki*

Shinshu University, Faculty of Engineering  
Department of Information Engineering  
Nagano-shi Wakasato 4-17-1, Nagano-ken 380-8553 JAPAN  
yamazaki@cs.shinshu-u.ac.jp

**Abstract** – In this paper, we define the equivalence of logic circuits and examine their description in VHDL. We can describe logic circuit components as relations between input and output signals in VHDL and implement such circuits on FPGA. In this case, it is necessary to verify the equivalence of VHDL descriptions and the actual logic circuits constructed based on these descriptions.

In a previous work, we proved the logical correctness of some circuits using the Mizar system. We also verified the equivalence of VHDL descriptions and the proved logic circuits.

**Keywords** – formalized mathematics, gate, logic circuit, VHDL.

## 1. Introduction

Recently, circuits are becoming very large-scale and high-speed, but their development periods are required to be short. Furthermore, in embedded systems, we often need circuits with special performance requirements corresponding to a particular purpose.

There are several methods for designing circuits using hardware description languages. Typical examples of such languages include VHDL (Very high-speed IC Hardware Designing), Verilog-HDL, and ABEL. Circuit structures and functions can be described using these languages and circuit simulations can be used to verify their performance. These verified circuits can be implemented in devices such as FPGA (field programmable gate array) or CPLD (complex programmable logic device) elements. Especially on FPGAs, designers can implement circuits by only describing their functions, without having to construct complex logic circuits. This improves development efficiency and development periods can be shortened. But, it is necessary to verify the equivalence of a hardware language description of a circuit and the circuit actually implemented in a device.

In this paper, we verify the correctness of gate circuits using Mizar and examine the equivalence of behavior level descriptions and gate circuits.

## 2. Logic Gates and Logic Equivalence of Adders

In this section we show a method for verifying the logical correctness of circuits. This notion is shown in Mizar article GATE1 [1]. In the article, basic gate logic elements are defined.

The main idea is to convert the specification of a logic circuit to Mizar propositional formulas. With this idea, we can use the strong inference abilities of the Mizar checker and the logical correctness of circuits can be easily proven by the Mizar system. The

Mizar checker treats more than 100 lines of formula and over 100 variables. This indicates that the Mizar system can prove the correctness of a middle scale logic circuit.

The policy in the article is as follows: when the functional (semantic) correctness of a system is already proven and the correspondence of the system to a (normal) logic circuit is given, it is enough to prove the correctness of the new circuit by proving the logical equivalence between them. By using this policy, we can easily apply the results to real problems.

The definitions of some basic gate logic elements are shown below.

**notation**

```
let a be set;
  antonym $a for a is empty;
end;
```

**definition**

```
let a be set;
  func NOT1 a equals
  {} if $a
  otherwise {{}: not contradiction};
end;
```

**definition**

```
let a, b be set;
  func AND2(a, b) equals
  NOT1 {} if $a & $b
  otherwise {};
end;
```

**definition**

```
let a, b be set;
  func OR2(a, b) equals
  NOT1 {} if $a or $b
  otherwise {};
end;
```

These are used to prove the equivalence of MSB carry operations of a 4-bit carry skip adder and the MSB carry of a normal 4-bit adder.

**theorem**

```
for c1,x1,x2,x3,x4,y1,y2,y3,y4,c2,c3,c4,c5,n1,n2,n3,n4,n,c5b being set holds
  ::Specification of Normal 4 Bit Adder
  ($c2 iff \$MAJ3(x1,y1,c1)) &
  ($c3 iff \$MAJ3(x2,y2,c2)) &
  ($c4 iff \$MAJ3(x3,y3,c3)) &
  ($c5 iff \$MAJ3(x4,y4,c4)) &
  ::Specification of Carry Skip Adder
  ($n1 iff \$OR2(x1,y1))&
  ($n2 iff \$OR2(x2,y2))&
  ($n3 iff \$OR2(x3,y3))&
  ($n4 iff \$OR2(x4,y4))&
  ($n iff \$AND5(c1,n1,n2,n3,n4))&
  ($c5b iff \$OR2(c5,n))
  implies ($c5 iff \$c5b);
```

In Mizar article [2], the authors proved the equivalence of the output of a 4-bit carry look ahead adder and the output of a normal 4-bit adder.

### 3. Correctness of Counter Circuits

In this section we show the correctness of counter circuits. These are basic logic circuits and can be constructed with the logic gates defined in the previous section.

Mizar article GATE2 [2] introduces the verification of the correctness for the operations and the specification of a 3-bit counter. Both the cases without reset input and the cases with reset input are considered. Mizar article GATE3 [3] introduces the verification of the correctness for the operations and specification of the Johnson counter. The authors formalize the concepts of 2-bit, 3-bit, and 4-bit Johnson counter circuits with reset inputs and define the specification of the state transitions without minor loops.

The next theorem shows the correctness of a 2-bit Johnson counter circuit without a minor loop.

```
theorem :: GATE_3:1 ::2JC:
  for s0,s1,s2,s3,ns0,ns1,ns2,ns3,q1,q2,nq1,nq2 being set holds
    ($s0 iff $AND2(NOT1 q2, NOT1 q1))&
    ($s1 iff $AND2(NOT1 q2, q1))&
    ($s2 iff $AND2(q2, NOT1 q1))&
    ($s3 iff $AND2(q2, q1)) &
    ($ns0 iff $AND2(NOT1 nq2,NOT1 nq1))&
    ($ns1 iff $AND2(NOT1 nq2, nq1))&
    ($ns2 iff $AND2(nq2,NOT1 nq1))&
    ($ns3 iff $AND2(nq2, nq1)) &
    ($nq1 iff $NOT1 q2)&
    ($nq2 iff $q1)
  implies
    ($ns1 iff $s0)&
    ($ns3 iff $s1)&
    ($ns2 iff $s3)&
    ($ns0 iff $s2);
```

The correctness of more complex Johnson counters are verified based on this theorem.

### 4. Hardware Description in VHDL

In this section, we consider hardware descriptions in VHDL. There are two ways to describe a circuit in VHDL. One is to describe it as a composite of gate circuits and the other is to describe it as functions of input signals. We consider the 1-bit full adder as an example.

## 4.1 Gate Level Description

Figure 1 shows the circuit diagram of a 1-bit full adder. Below this is a sample of a gate level description of the 1-bit full adder. In this case, it is easy to understand and imagine its actual gate circuit, but it is very difficult to do this when designing a complex circuit.

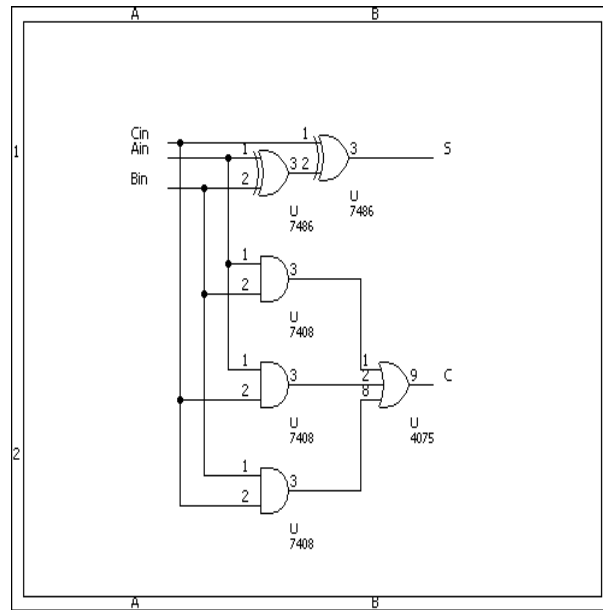


Figure 1. Circuit diagram for 1-bit full adder.

```
begin -- RTL
  S <= C_IN xor (A_IN xor B_IN);
  C <= (A_IN and B_IN) or (A_IN and C_IN) or (B_IN and C_IN);
end RTL;
```

Figure 2. Gate level description of 1-bit full adder circuit in VHDL.

This description is the same as the logic circuit shown in the previous section and the equivalence of each description is easily verified.

## 4.2 Behavior Level Description

When we use behavior level descriptions, we need to consider the responses against input signals. The relations of input signals and output signals are shown as a truth table. Table 1 is a truth table for the 1-bit full adder.

**Table 1.** Truth table of 1-bit full adder

Cin	Ain	Bin	S	C
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

The code in Fig. 3 is the behavior level description of a 1-bit full adder based on the truth table in VHDL.

```
begin -- RTL
  -- truth_table:
  if (X_in = "0" and Y_in = "0" and C_in = "0") then S_out <= "0"; C_out <= "0";
  elsif (X_in = "0" and Y_in = "1" and C_in = "0") then S_out <= "1"; C_out <=
"0";
  elsif (X_in = "1" and Y_in = "0" and C_in = "0") then S_out <= "1"; C_out <=
"0";
  elsif (X_in = "1" and Y_in = "1" and C_in = "0") then S_out <= "0"; C_out <=
"1";
  elsif (X_in = "0" and Y_in = "0" and C_in = "1") then S_out <= "1"; C_out <=
"0";
  elsif (X_in = "0" and Y_in = "1" and C_in = "1") then S_out <= "0"; C_out <=
"1";
  elsif (X_in = "1" and Y_in = "0" and C_in = "1") then S_out <= "0"; C_out <=
"1";
  elsif (X_in = "1" and Y_in = "1" and C_in = "1") then S_out <= "1"; C_out <=
"1";
  end if;
  -- truth_table;
end RTL;
```

Figure 3. Behavior level description of 1-bit full adder in VHDL.

In this case, we do not need to construct a gate level circuit and it is easy to describe a complex circuit from the specification. Therefore, the development time required is shorter.

## 5. Verification of Behavior Level Description by Mizar

It is popular to create circuits using behavior level descriptions because it makes developing a circuit easy. But, a circuit implemented from a VHDL description is composed from gate circuits. Therefore, it is necessary to verify the equivalence between behavior level descriptions and their corresponding gate circuits.

The next theorem verifies the equivalence of a 1-bit full adder behavior level description in VHDL and an actual 1-bit full adder gate circuit by Mizar.

```

theorem :: 1-bit full adder
  ((not $x & not $y & not $ci) implies (not $s & not $co)) & :: (0, 0, 0) => (0, 0)
  (
    not (not $x & not $y & not $ci) implies
      (((not $x & $y & not $ci) implies ($s & not $co)) & :: (0, 1, 0) => (1, 0)
      (
        not (not $x & $y & not $ci) implies
          (((($x & not $y & not $ci) implies ($s & not $co)) & :: (1, 0, 0) => (1, 0)
          (
            not ($x & not $y & not $ci) implies
              (((($x & $y & not $ci) implies (not $s & $co)) & :: (1, 1, 0) => (0, 1)
              (
                not ($x & $y & not $ci) implies
                  (((not $x & not $y & $ci) implies ($s & not $co)) & :: (0, 0, 1) => (1, 0)
                  (
                    not (not $x & not $y & $ci) implies
                      (((not $x & $y & $ci) implies (not $s & $co)) & :: (0, 1, 1) => (0, 1)
                      (
                        not (not $x & $y & $ci) implies
                          (((($x & not $y & $ci) implies (not $s & $co)) & :: (1, 0, 1) => (0,
1)
                          (
                            not ($x & not $y & $ci) implies
                              ((($x & $y & $ci) implies ($s & $co)) :: (1, 1, 1) => (1, 1)
                              ))))))))))))
      ))))))))
  )
iff
  (
    (($s iff $XOR2(ci, XOR2(x, y))) & :: s
    ($co iff $OR3(AND2(x, y), AND2(x, ci), AND2(y, ci)))) :: c
  );

```

In this case, we show only the 1-bit full adder case.

The n-bit full adder equivalence is easily verified by combining 1-bit full adder circuits.

## 6. Conclusions

We reported a method for defining logic gates and the correctness of some sample circuits. We also verified the equivalence of a circuit and its behavior level description in VHDL.

For our example, we verified the equivalence of a 1-bit full adder circuit described in VHDL and its corresponding gate circuit. We plan to verify more complicated circuits, for example circuits with feedback signals. By guaranteeing the equivalence of circuits designed by VHDL and their logic circuits, we can make development periods significantly shorter and it will be easy for designers to make circuits for special purposes and embedded systems.

## References

- [1] Y. Nakamura, *Logic Gates and Logical Equivalence of Adders*, Formalized Mathematics, Vol. 8, No. 1, 1999, pp.35-45.
- [2] Y. Yang, K. Wasaki, Y. Fuwa, and Y. Nakamura, *Correctness of Binary Counter Circuits*, Formalized Mathematics, Vol. 8, No. 3, 1999, pp.83-86.
- [3] Y. Yang, K. Wasaki, Y. Fuwa, and Y. Nakamura, *Correctness of Johnson Counter Circuits*, Formalized Mathematics, Vol. 8, No. 1, 1999, pp.87-92.
- [4] H. Yamazaki and K. Wasaki, *The Correctness of the High Speed Array Multiplier Circuits*, Formalized Mathematics, Vol. 9, No. 3, 2001, pp. 475-479.

