Verification of logic circuits using Mizar and its application to an adder circuit on a radix-$2^k$SD number.

1. Introduction

To answer the request of higher performance from electric equipment, logic circuit is becoming more complicated and more large-scaled. This makes it more difficult to verify correctness of a designed circuit.

Heretofore, to verify a logical circuit, output for every possible state and input should be confirmed by simulation. However, when the scale of a circuit increases, states of circuit increases exponentially, accordingly (Example, when a circuit has 100 return wires, it has $2^{100}$ states.). So it is impossible or very difficult to complete such a simulation for large-scaled circuits.

As a new way to verify correctness of logical circuit, we express logical circuit with mathematical description (called mathematical model). The correctness of the circuits is assured when correctness of the
A proof checker system has been used to verify correctness of proof in mathematics. It can be applied to verifying correctness of a designed circuit with the method we suggest.

In this paper, after introducing the proof checker system Mizar which is used to verify correctness of proof (Section 2), we give some definitions as a preparation for mathematics descriptions of a logical circuit (Section 3). Then, we explain how a logical circuit is described by these definitions, and how its correctness is verified by the system (Section 4). At last, as an example, we apply the method in designing the adder circuit on a radix-2^kSD number.

2. Proof checker system Mizar

As an attempt to reconstruct mathematical vernacular, the Mizar project started in 1973. [1]

And, it has become the most important activity in the project to develop the database of mathematics since 1989. Now, more than 2,000 definitions and 20,000 theorems are included in the increasing database.

As a characteristic of Mizar, useful verified proof is accepted by Mizar's library. Using Mizar, besides mathematical proof, a mathematical model can be verified too.

Just as a large-scale circuit can be designed as a combination of
smaller circuits which function has been verified, the correctness of a model can be showed when the model is a combination of smaller models which has been accepted by the library.

3. Preparation for mathematical description of a logical circuit

We give a relation between basic concepts of logical circuits and mathematics as follows:

(1) We think input and output signals as sets. The logic of a signal line has 2 states, 0 and 1. 0 is defined as the empty set \( \emptyset \), and 1 is defined as a non-empty set.

It is described as follows at Mizar:

\[
\text{definition let } a \text{ be set;}
\text{ redefine attr } a \text{ is empty;}
\text{ antonym } Sa; \\
\text{ end;}
\]

(2) We consider the expression of every possible states formed by input and output signals. It is described like this:

\[
S = \emptyset \text{ iff } S = \text{AND}\{\text{NOT}(q_2), \text{NOT}(q_1)\}
\]

(3) We define a circuit as a Boolean function of sets defined above.

For example, NOT circuit writes follows:

\[
\text{func NOT}(a) \rightarrow \text{set equals}
\phi \text{ if } Sa \\
\text{ otherwise } \{\phi \text{ : not contradiction}\}; \\
\text{ end;}
\]
4. A new method of logic circuit's verification using Mizar system

Here, we introduce how the new method works with a simple example.

Consider a 3bit up counter circuit. The correctness of the circuit can be guaranteed with Mizar at the following steps.

(1) Describing the input and output as sets.
(2) Defining every possible state of input and output as following with the set of step 1. (Fig.1)

\[ \text{$s0 = \text{AND3(NOT1 } q3, \text{ NOT1 } q2, \text{ NOT1 } q1);} \]
\[ \text{$s1 = \text{AND3(NOT1 } q3, \text{ NOT1 } q2, \text{ NOT1 } q1);} \]
\[ \text{$s2, s3, s4, s5, s6, s7$ is similar to $s0, s1$.} \]

(3) Expressing the behavior of 3bit up counter circuit with Boolean expressions as follows: (Fig.2)

\[ \text{$nq1 = \text{AND2(NOT1 } q1, \text{ R})} \]
\[ \text{$nq2 = \text{AND2(XOR2(q1, q2), R)})} \]
\[ \text{$nq3 = \text{AND2(OR2(AND2(q3, NOT1 q1), AND2(q1, XOR2(q2, q3))), R)})} \]

Here, $q1, q2, q3, R$ are circuit inputs and $nq1, nq2, nq3$ are outputs.

(4) Verifying the correctness of circuit by confirming its Boolean expressions' tautology using Mizar system.

(\(S_{n0} \iff \text{SAND2}(s0, R)\)) & (\(S_{n1} \iff \text{SAND2}(s1, R)\)) & (\(S_{n2} \iff \text{SAND2}(s2, R)\)) & (\(S_{n3} \iff \text{SAND2}(s3, R)\)) & (\(S_{n4} \iff \text{SAND2}(s4, R)\)) & (\(S_{n5} \iff \text{SAND2}(s5, R)\)) & (\(S_{n6} \iff \text{SAND2}(s6, R)\)) & (\(S_{n7} \iff \text{SOR2}(s7, NOT1 R)\));

Here, $s0, \ldots, s7$ means current states and $S_{n0}, \ldots, S_{n7}$ means next states of current states.
(\(\text{sns1 iff } \text{SAND2}(s0, R)\)) means the next state will be \(\text{sns1}\), if and only if the current state is \(s0\) and \(R\) is "1".

Behavior of a 3bit up counter circuit
Have 4 inputs (\(R, q3, q2, q1\)) and 3 outputs (\(nq3, nq2, nq1\))
States change as follows:
\[000 \rightarrow 001 \rightarrow 010 \rightarrow 011 \rightarrow 100 \rightarrow 101 \rightarrow 110 \rightarrow 111 \rightarrow 000 \rightarrow\]
return to the initial state (000) by the reset input

Signal definitions

![Diagram of 3bit up counter]

Input signal definitions
\(000 \rightarrow \text{sns0 = SAND3(NOT1 q3, NOT1 q2, NOT1 q1)}\)
Output signal definitions
\(001 \rightarrow \text{sns1 = SAND3(q3, NOT1 q2, q1)}\)

Fig. 1 Definitions of 3bit up counter.

Definitions and proof to correctness of a 3bit up counter

![Diagram of 3bit up counter]

Needs to proof these definitions
\(\text{sns1 iff } \text{SAND2}(s0, R)\) &
\(\text{sns2 iff } \text{SAND2}(s1, R)\) &
\(\text{sns3 iff } \text{SAND2}(s2, R)\) &
\(\text{sns4 iff } \text{SAND2}(s3, R)\) &
\(\text{sns5 iff } \text{SAND2}(s4, R)\) &
\(\text{sns6 iff } \text{SAND2}(s5, R)\) &
\(\text{sns7 iff } \text{SAND2}(s6, R)\) &
\(\text{sns0 iff } \text{SOR2}(s7, \text{NOT1 R})\);

Definitions of a 3bit up counter
\(\text{sns1 iff } \text{SAND2}(\text{NOT1 q1}, R)\) &
\(\text{sns2 iff } \text{SAND2}(\text{XOR2(q1, q2)}, R)\) &
\(\text{sns3 iff } \text{SAND2}(\text{OR2(AND2(q3, \text{NOT1 q1)}), AND2(q1, XOR2(q2, q3))}, R)\)

Fig. 2 Definitions (cont.) and proof of correctness of 3bit up counter.
5. An application to a radix-$2^k$SD number coded adder circuit

Here, we apply the new method to designing an adder circuit on a radix-$2^k$SD number.

In a radix-$2^k$SD (signed-digit) coded adder circuit, calculations can be finished in a constant time no matter whether there is a ripple carry.

Here, we will verify the correctness of such a circuit of case $k=2$.

A designed radix-4SD number circuit and signal layouts

Fig. 3 Signal layout of radix-4SD number coded adder circuit
Outputs of PART1 which used radix-4SD number

<table>
<thead>
<tr>
<th>Value</th>
<th>w2 w1 w0 nc1 nc0</th>
<th>other expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>-6</td>
<td>-4 + -2</td>
<td>1 1 0 1 1</td>
</tr>
<tr>
<td>-5</td>
<td>-4 + -1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>-4</td>
<td>-4 + 0</td>
<td>0 0 0 1 1</td>
</tr>
<tr>
<td>-3</td>
<td>-4 + 1</td>
<td>0 0 1 1 1</td>
</tr>
<tr>
<td>-2</td>
<td>0 + -2</td>
<td>1 1 0 0 0</td>
</tr>
<tr>
<td>-1</td>
<td>0 + -1</td>
<td>1 1 1 0 0</td>
</tr>
<tr>
<td>0</td>
<td>0 + 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 + 1</td>
<td>0 0 1 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 + 2</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td>3</td>
<td>4 + -1</td>
<td>1 1 1 0 1</td>
</tr>
<tr>
<td>4</td>
<td>4 + 0</td>
<td>0 0 0 0 1</td>
</tr>
<tr>
<td>5</td>
<td>4 + 1</td>
<td>0 0 1 0 1</td>
</tr>
<tr>
<td>6</td>
<td>4 + 2</td>
<td>0 1 0 0 1</td>
</tr>
</tbody>
</table>

Several kinds numerical value expression is possible so that figure increasing number can take out a mark in SD number.

In this study, we represent numerical value by expression surrounded by a square.

Fig.4 Definitions of radix-4SD number coded adder circuit

The correctness of a 4-SD number adder circuit is verified with the 4 steps described in former section.

First, input and output status can be defined as follows:

INPUT STATE:

($x_0,$ $x_1,$ $x_2$) = (NOT1 $x_1$, $x_0$) & ($x_2,$ NOT1 $x_1$, NOT1 $x_0$) &
($x_1,$ NOT1 $x_1$, $x_0$) &
(NOT1 $x_2,$ NOT1 $x_1$, NOT1 $x_0$) &
(NOT1 $x_2,$ NOT1 $x_1$, $x_0$) &
(NOT1 $x_2,$ NOT1 $x_1$, $x_0$) &
(NOT1 $x_2,$ NOT1 $x_1$, $x_0$) &

Here, $x_0,$ $x_1,$ $x_2$ express the three inputs of PART1 (Fig.3), $x_0,$ $x_1,$ $x_2$ are all possible input states. The expression $x_0$ iff SAND3($x_2,$ NOT1 $x_1$, NOT1 $x_0$) means that an input state is called $x_0$ if and only if inputs $x_2='1'$, $x_1='0'$, $x_0='1'$.
Sym3, Sym2, Sym1, Syz, Syp1, Syp2, Syp3 can be defined similarly.

**OUTPUT STATE:**

Output states can be defined in same way as follows.

\[(Snz \text{ iff } SAND5(NO{1n}c1, NO{1n}c0, NO{1n}l1, NO{1n}l0)) \\
(Snp1 \text{ iff } SAND5(NO{1n}c1, NO{1n}c0, NO{1n}l1, NO{1n}l0)) \\
(Snp2 \text{ iff } SAND5(NO{1n}c1, NO{1n}c0, NO{1n}l1, NO{1n}l0)) \\
(Snp3 \text{ iff } SAND5(NO{1n}c1, NO{1n}c0, NO{1n}l1, NO{1n}l0)) \\
(Snp4 \text{ iff } SAND5(NO{1n}c1, NO{1n}c0, NO{1n}l1, NO{1n}l0)) \\
(Snp5 \text{ iff } SAND5(NO{1n}c1, NO{1n}c0, NO{1n}l1, NO{1n}l0)) \\
(Snp6 \text{ iff } SAND5(NO{1n}c1, NO{1n}c0, NO{1n}l1, NO{1n}l0))
\]

Next, the behavior of PART1 can be described as a Boolean function as follows.

\[(Sn0 \text{ iff } SOR8(AND4(NO{1n}x2, NO{1n}x1, NO{1n}y2, NO{1n}y1)), \]
\[\text{AND3(NO{1n}x2, NO{1n}y2, OR2(AND2(x1, x0), AND2(y1, y0))),}\]
\[\text{AND3(NO{1n}x2, NO{1n}y2, OR2(AND2(x1, y0), AND2(x0, y1))),}\]
\[\text{AND3(NO{1n}x1, NO{1n}y1, OR2(AND4(x2, x0, NO{1n}y2, NO{1n}y0),}\]
\[\text{AND4(NO{1n}x2, NO{1n}x1, y2, y0))),}\]
\[\text{AND5(x2, x1, y2, y1, NAND2(x0, y0)), AND5(x2, x1, y2, NO{1n}y1, y0),}\]
\[\text{AND5(x2, NO{1n}x1, x0, y2, y1), AND6(x2, NO{1n}x1, x0, y2, NO{1n}y1, y0))}\]

\$(Sn0, Snw2, Snw1, Snw0$ can be described in same way.

Then, the relation between input status and output status is built.

The following relation expression is showed tautology by Mizar system. So the correctness of PART1 circuit is verified. (Fig.5)

\[(Sn6 \text{ iff } SAND2(xm3, ym3)) \& \]
\[(Sn5 \text{ iff } SOR2(AND2(xm3, ym2), AND2(xm2, ym3))) \& \]
\[(Sn4 \text{ iff } SOR3(AND2(xm3, ym1), AND2(xm2, ym2), AND2(xm1, ym3))) \& \]
\[(Sn3 \text{ iff } SOR4(AND2(xm3, yz), AND2(xm2, ym1), AND2(xm1, ym2),\]
\[\text{AND2(xz, ym3))) \&}\]
\[(Sn2 \text{ iff } SOR5(AND2(xm3, yp1), AND2(xm2, yz), AND2(xm1, ym1),\]
\[\text{AND2(xz, ym2), AND2(xp1, ym3))) \&}\]
\[(Sn1 \text{ iff } SOR6(AND2(xm3, yp2), AND2(xm2, yp1), AND2(xm1, yz),\]
\[\text{AND2(xz, ym1), AND2(xp1, ym2), AND2(xp2, ym3))) \&}\]
Here, for example, the expression \( \text{SNm5} \) means a state is called \( \text{SNm5}(-5) \) if and only if the input state \( \text{SNm3(x=-3) and SNm2(y=-2)} \) or \( \text{SNm2(x=-2) and SNm3(y=-3)} \).

Other expressions are similar.

The correctness of PART2 can be verified in same way. (Fig.6)

Thus, the behavior of the whole circuit can be expressed by the following expressions.

\[
\begin{align*}
(\text{SNm2} &\iff \text{SNAND}(x2, x1, \text{NO1} x0)) &
(\text{SNm1} &\iff \text{SNAND}(x2, x1, x0)) &
(\text{SNz} &\iff \text{SNAND}(\text{NO1} x2, \text{NO1} x1, \text{NO1} x0)) &
(\text{SNp1} &\iff \text{SNAND}(\text{NO1} x2, \text{NO1} x1, x0)) &
(\text{SNp2} &\iff \text{SNAND}(\text{NO1} x2, x1, \text{NO1} x0)) &
(\text{SNc} &\iff \text{SNAND}(c1, c0)) &
(\text{SNz} &\iff \text{SNAND}(\text{NO1} c1, \text{NO1} c0)) &
(\text{SNp} &\iff \text{SNAND}(\text{NO1} c1, c0)) &
(\text{SNm3} &\iff \text{SNAND}(\text{NO1} ns2, \text{NO1} ns1, ns0)) &
(\text{SNm2} &\iff \text{SNAND}(\text{NO1} ns2, ns1, \text{NO1} ns0)) &
(\text{SNm1} &\iff \text{SNAND}(\text{NO1} ns2, ns1, ns0)) &
(\text{SNz} &\iff \text{SNAND}(\text{NO1} ns2, \text{NO1} ns1, \text{NO1} ns0)) &
(\text{SNp1} &\iff \text{SNAND}(\text{NO1} ns2, \text{NO1} ns1, ns0)) &
\end{align*}
\]
The tautology can be showed because part 1 and part 2 have been verified. So the correctness of radix-4SD adder circuit is verified.
Fig. 5 The circuit verified of correctness by Mizar system (PART 1)

nc1 = x2 * ~x1 * ~x0 + y2 * ~y1 * ~y0
+ ~x2 * ~x1 * ~x0 * y2 * ~y1 + x2 * ~x1 * ~y2 * ~y1 * ~y0
+ x2 * y2 * (~x1 + ~y1) + x2 * y2 * (x1 * ~x0 + y1 * ~y0)
nc0 = ~x2 * ~y2 * (x1 * x0 + y1 * y0)
+ ~x2 * ~y2 * (x1 * y0 + x0 * y1) + ~x2 * x1 * ~y2 * y1
+ ~x1 * ~y1 * (x2 * x0 * ~y2 * ~y0 + ~x2 * ~x0 * y2 * y0)
+ x2 * x1 * y2 * ~y1 * y0 + x2 * ~x1 * x0 * y2 * y1
+ x2 * ~x1 * x0 * y2 * ~y1 * y0
w2 = ~x2 * ~y2 * (x1 XOR y1) * (x0 XOR y0)
+ x2 * x1 * ~y2 * ~y1 * (x0 NAND y0)
+ ~x2 * ~x1 * y2 * y1 * (x0 NAND y0)
+ x2 * ~x1 * y1 * ~y0 + x1 * ~x0 * y2 * ~y1
+ ~x1 * x0 * ~y1 * y0 * (x2 XOR y2)
+ x2 * x0 * y2 * y0 * (x1 XNOR y1)
w1 = (x1 XNOR y1) * (x0 * y0) + (x1 XOR y1) * (x0 NAND y0)
w0 = x0 XOR y0

Fig. 6 The circuit verified of correctness by Mizar system (PART 2)

s2 = c1 * ~c0 + ~w2 * ~w1 * ~w0 * c1
+ ~w2 * w1 * w0 + w2 * w1 * ~w0
+ x2 * ~x1 + x2 * ~c1 * ~c0 + x2 * c1
s1 = ~w2 * ~w1 * ~w0 * c1 * c0
+ ~w2 * ~w1 * w0 * ~c1 * c0
+ w1 * ~w0 * ~c1
+ w2 * w1 * w0 * c1 * c0
+ x2 * x1 * x0 * ~c1 * ~c0
s0 = ~w2 * ~w1 * ~w0 * c0 + x1 * ~x0 * c0
+ ~x2 * ~x1 * x0 * ~c1 * ~c0
+ x2 * x1 * x0 * ~c1 * ~c0
6. Conclusion

We showed it is possible to verify correctness of logical circuit’s mathematical model using proof checker Mizar.

This can be considered as a new approach to verifying correctness of logical circuit.

The circuit we proved has been accept by the library of Mizar, and it can be used to prove larger circuits.

When the library is substantially in future, verification of logical circuits in practice can be expected. Verification of cryptogram circuit can be realized in same way.

References

