Concurrent Error Detection by Complementing Inputs

Center for Reliable Computing

Concurrent Error Detection by Complementing Inputs

by

Nahmsuk Oh

RATS

Feb., 23, 1998

Outline
- Motivation
- Introduction to error detection
- Overcome the disadvantage of stutter step (repetition)
- Duality
- Addition
- Summary

Motivation
- Is this calculation correct?
- How are you going to check if it is correct or not?

Stutter Step Mode (repetition)
- Repeat the computation and compare it with the original result.
  \[ c = a + b \]
  \[ d = a + b \]
  \[ \text{if } (c \neq d) \text{ error} \]

Multitasking Environment
- Two or three processes doing same jobs:
  Compare their results

Drawback - Time Redundancy
- The error that lasts long may not be detected.
- Permanent errors may not be detected.
Duality - EE275 / CS211

- Metatheorem
  Any true theorem about switching algebra whose statement involves only the three operations +, •, and ° remains true if (+ and •) and (0 and 1) are interchanged throughout.

- Definition
  Given a Boolean expression E(x₁, x₂, ..., xₙ, •, +, °), the dual of E is defined as
  dual of E = D[E] = E(x₁, x₂, ..., xₙ, +, •, °)

Duality - example (1)
- DeMorgan’s theorem
  f'(x₁, x₂, ..., xₙ) = D[f(x₁', x₂', ..., xₙ')]
  f(x, y, z) = xy' + z
  f(x, y, z)' = (x' + y)z'
  f(x', y', z') = x'y + z'
  D[f(x', y', z')] = (x' + y)z'

Duality - example (2)
- Stutter step with dual operation
  Repeat the execution but with the complemented inputs and dual operators
  example)
  and c a b
  or d a’ b’
  compare c d’

Duality - example (2)

1010 AND 0011 OR 1100
0010 1101

0101 stuck at 1

1111 AND 0011 OR 1100
0011 1101

0010

Duality - example (3)
- a + (or) b --> (a'b')'
  ab --> (a' + b')'
  a • b --> a' • b'
- addition?
  a + b --> ?

Advantage
- Time redundancy
  - no hardware redundancy
- Detect errors that may last long
- Detect permanent errors
Concurrent Error Detection by Complementing Inputs

Addition
- At the first step, addition with original data.
- At the second step, addition with two's complement data (e.g., negative numbers).
- If their absolute values match, the result is correct.
- If an error modifies the result, the error is detected.

Addition - example(1)
- If an error modifies the result, the error is detected.

\[
\begin{array}{c}
0010 \\
+ 0011
\end{array}
\begin{array}{c}
1110 \\
+ 1101
\end{array}
\begin{array}{c}
0101 \\
+ 1011
\end{array}
\begin{array}{c}
0101 \\
\text{stuck at 1}
\end{array}
\begin{array}{c}
0011 \\
+ 0011
\end{array}
\begin{array}{c}
1111 \\
+ 1101
\end{array}
\begin{array}{c}
0110 \\
+ 1100
\end{array}
\begin{array}{c}
0101
\end{array}
\]

Addition - example(2)
- If an error does not modify the result, the error is not detected, however, the result is correct.

(Fault neutralization)

\[
\begin{array}{c}
0010 \\
+ 0011
\end{array}
\begin{array}{c}
1110 \\
+ 1101
\end{array}
\begin{array}{c}
0101 \\
\text{stuck at 0}
\end{array}
\begin{array}{c}
0010 \\
+ 0011
\end{array}
\begin{array}{c}
1110 \\
+ 1101
\end{array}
\begin{array}{c}
0101 \\
+ 1011
\end{array}
\begin{array}{c}
0101
\end{array}
\]

Adder
- Define:
  \[G_i = a_ib_i\] carry generation
  \[P_i = a_i \bar{b_i}\] carry propagation
- An adder is represented as:
  \[s_i = a_i \bar{b_i} \bar{c_i}\]
  \[c_{i+1} = G_i + P_ic_i\]

Error Model in Adder
- Error in carry logic \([c_{i+1} = G_i + P_ic_i]\)
- Error in bit sum logic \([s_i = a_i \bar{b_i} \bar{c_i}]\)

Error in Carry Logic \([c_{i+1} = G_i + P_ic_i]\)
- Model the error by adding/subtracting \(2^m\) to/from the original input data. (\(m\) is the \(m\)th stage where error occurred)

(example)
Assume a carry is generated at \(m\)th stage of adder because of stuck at 1. Model it as:
\[
\begin{align*}
\text{a} + \text{b} + \text{b} + 2^m & \quad -\text{a} - \text{b} + 2^m \\
\text{a} + \text{b} & \quad -\text{a} + \text{b} - 2^m
\end{align*}
\]
Concurrent Error Detection by Complementing Inputs

Error in Bit Sum Logic ($s_i = a_i \oplus b_i \oplus c_i$)

- stuck at 1 in $m$th bit of input data (0->1): $+2^m$
- stuck at 0 in $m$th bit of input data (1->0): $-2^m$

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$b_i$</th>
<th>$c_i$</th>
<th>$s_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>2</td>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0110</td>
<td>-2</td>
<td>1110</td>
<td>-2</td>
</tr>
<tr>
<td>1101</td>
<td>-3</td>
<td>1101</td>
<td>-3</td>
</tr>
<tr>
<td>1011</td>
<td>-5</td>
<td>1011</td>
<td>-5</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>1001</td>
<td>9</td>
</tr>
</tbody>
</table>

Example - FFT

```c
for (m = 1; m < mmax; m += 2) {
    for (i = m; i >= n; i += istep) {
        j = i + mmax;
        j_ = (i_ + mmax_);
        temp_ = -(wr_ * data[-j_] - wi_ * data[-j_ + 1]);
        temp_ = -(wr_ * data[-j_ + 1] + wi_ * data[-j_]);
        data[m][-j_] = data[m][-i_] - temp;
        data[m][-j_ + 1] = data[m][-i_ + 1] - temp;
        data[m][-i_] += temp;
        data[m][-i_ + 1] += temp;
    }
    wr = (temp = wr) * wpr - wi * wpi + wr;
    wi = wi * wpr + temp * wpi + wi;
    mmax = istep;
}
```

Example - FFT with error detection

```c
m_ = 1;
for (m = 1; m < mmax; m += 2) {
    m_ += -2;
    /*printf( "m_ =%d  mmax_=%d\n", m_, mmax_); */
    if (m_ < mmax_) error(1);
    i_ = m_ - istep_
    for (i = m; i >= n; i += istep) {
        i_ += istep;
        if (i_ < = n) error(2);
        j = i + mmax;
        j_ = (i_ + mmax_);
        temp = wr * data[j] - wi * data[j + 1];
        temp = -(wr_ * data[-j_] - wi_ * data[-j_ + 1]);
        temp = -(wr_ * data[-j_ + 1] + wi_ * data[-j_]);
        data[j] = data[i] - temp;
        data[m][-j_] = data[m][-i_] - temp;
        data[j + 1] = data[i + 1] - temp;
        data[m][-j_ + 1] = data[m][-i_ + 1] - temp;
        data[i] += temp;
        data[m][-i_] += temp;
        data[i + 1] += temp;
        data[m][-i_ + 1] += temp;
        mexpr = error(data[i], data[i], 4);
    }
    wr = (temp = wr) * wpr - wi * wpi + wr;
    wr_ = -(-(temp = wr_) * (-wpr_) - (-wi_) * (-wpi_) + (-wr_));
    wi = wi * wpr + temp * wpi + wi;
    wi_ = -(-(wi_)*(-wpr_) + (-temp)*(-wpi_) + (-wi_));
    mexpr = error(wr, wr_, 1);
    mexpr = error(wi, wi_, 1);
    mmax_ = istep_;
}
```

Summary

- Error detection by using duality
- Error detection by applying complemented input data
- Detect transient as well as permanent error in ALU

References