The Design and Proof of Correctness of a Fault-Tolerant Circuit

William R. Bevier
William D. Young

Computational Logic, Inc.
1717 W. 6th Street
Austin, Texas

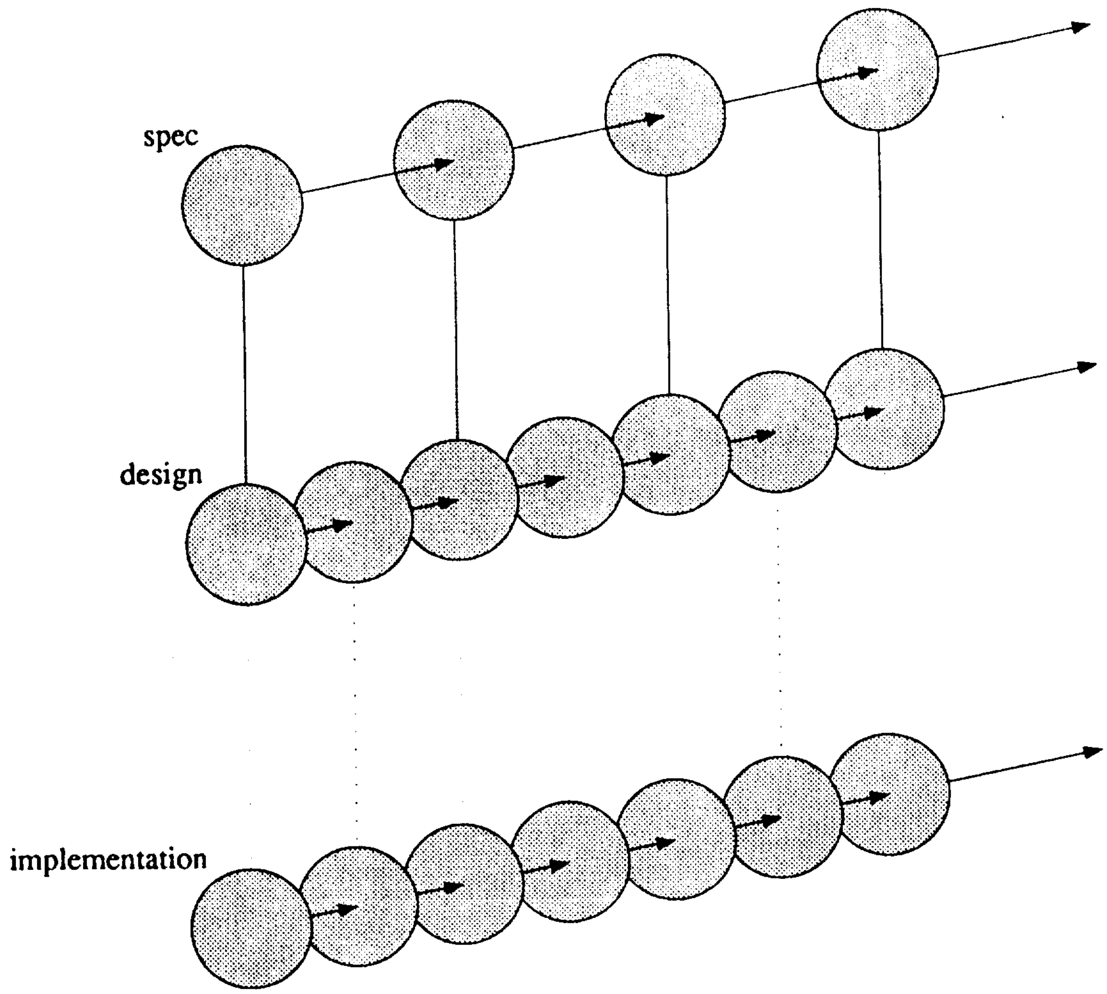
18 August 1990

What We Accomplished

- A formal statement of Interactive Consistency Conditions[1] in the Boyer-Moore logic.
- A formal statement of the Oral Messages algorithm *OM* in the Boyer-Moore logic.
- A mechanically checked proof that *OM* satisfies the Interactive Consistency conditions.
- A mechanically checked proof of the optimality result: no algorithm can tolerate fewer faults than *OM* yet still achieve Interactive Consistency.
- The use of *OM* in a functional specification for a fault-tolerant device.
- A formal description of the design of the device.
- A mechanically checked proof that the device design satisfies the specification.
- An implementation of the design in programmable logic arrays.

[1]See "The Byzantine Generals Problem", Lamport, Shostak and Pease, ACM Toplas, Vol 4, No 3, July 1982.

A Stack of Related Machines



The Specification

The specification is a function that describes a finite state machine.

At every step, each of N processes

1. reads its sensor input,

2. exchanges its sensor value with all other processes,

3. produces an *interactive consistency vector* (ICV) that contains what it concludes is each other process's value, and

4. applies a filter function to the ICV to produce an output.

Properties of the Specification Function

The exchange of sensor values is accomplished by an algorithm called *OM*.

OM achieves *interactive consistency*. That is,

A process sends a message to *n-1* destination processes.

1. All non-faulty destination processes agree on the same received value.
2. If the sending process is non-faulty, then every non-faulty destination process receives the message sent.

OM has been defined as a function in the Boyer-Moore logic, and a proof that interactive consistency is achieved has been mechanically checked.

Formal Statement of Correctness of *OM*

Let

- n be the number of processes,
- L be the set $\{0, ..., n-1\}$,
- $g, i, j \in L$ be process names,
- x be g's local value, and
- m give the number of rounds of information exchange.

The interactive consistency conditions are stated as follows.

$$\begin{aligned}
&\neg faulty(i)\\
&\&\neg faulty(j)\\
&\& 3 \cdot faults(L) < n\\
&\& faults(L) \le m\\
\rightarrow\\
&OM(n, g, x, m)[i] = OM(n, g, x, m)[j].
\end{aligned}$$

$$\begin{aligned}
&\neg faulty(g)\\
&\&\neg faulty(i)\\
&\& 3 \cdot faults(L) < n\\
&\& faults(L) \le m\\
\rightarrow\\
&OM(n, g, x, m)[i] = x
\end{aligned}$$

Specification Abstraction

The following aspects of the specification are not constrained:

1. The number of processes.

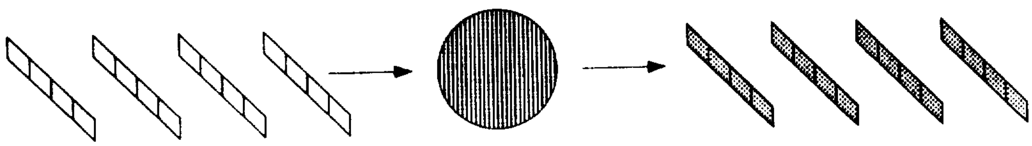
2. The types of the input and output values.

3. The nature of the filter function.

What Interactive Consistency Guarantees

The specification can be thought of as a function which

- receives a sequence of N-tuples of input values, and
- produces a sequence of N-tuples of output values.

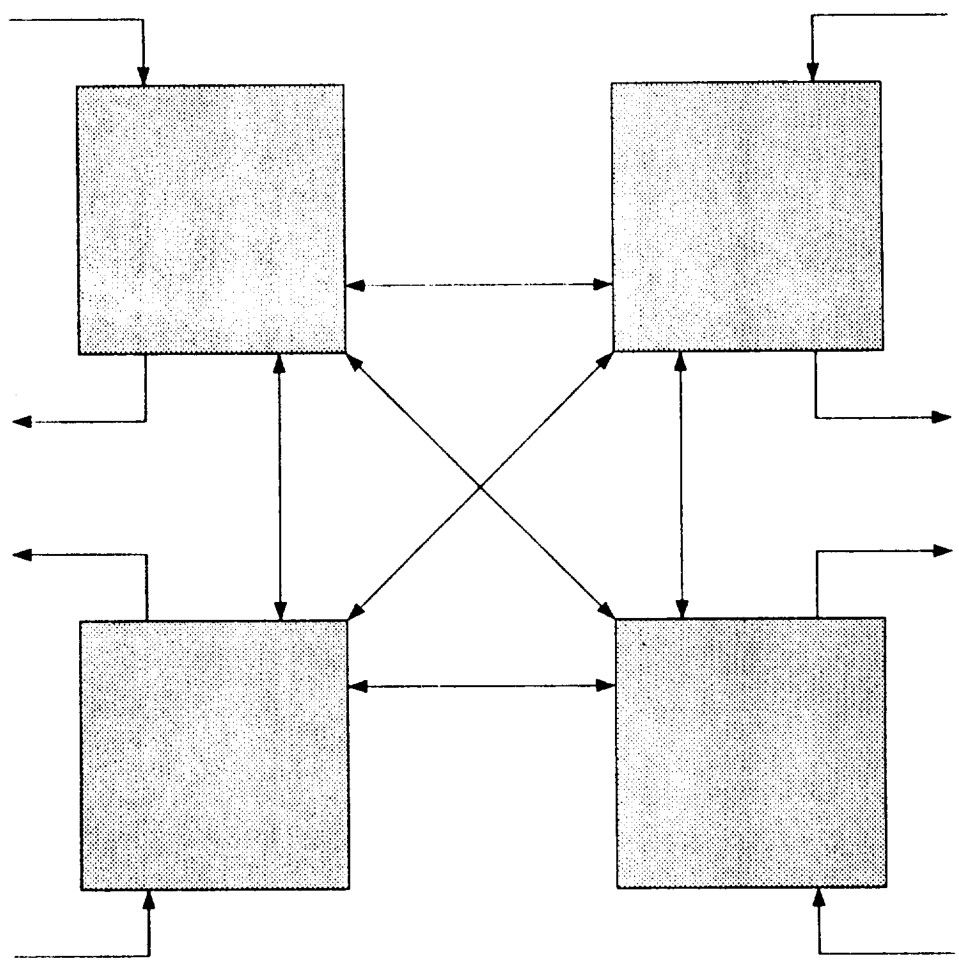


Because of Interactive Consistency, we can conclude:

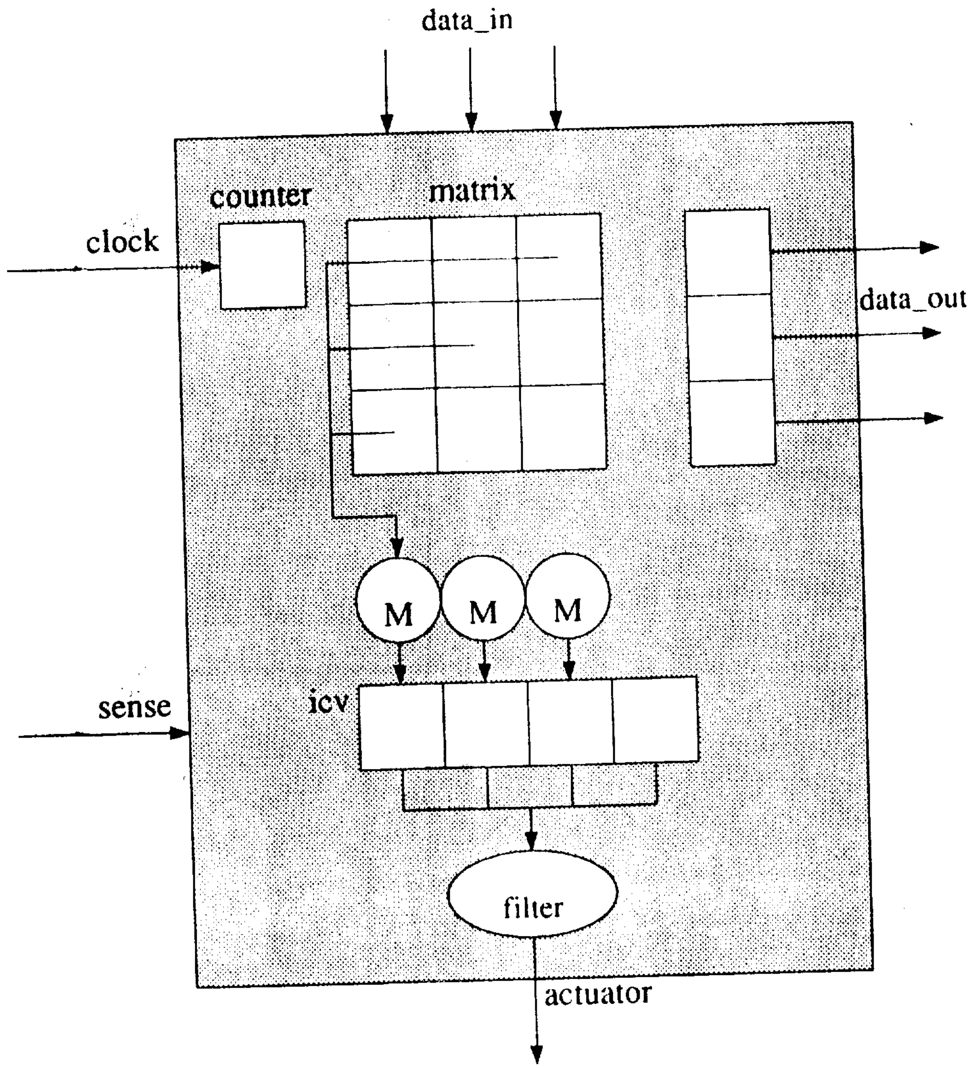
At each step, all non-faulty processes agree on their output iff the total number of processors exceeds three times the number of faulty processors.

The Device Design

Goal: Design 4 identical circuits which, when operating synchronously, achieve Byzantine agreement.



A Process Internal State



Process Steps

```
0: data_out[i]    ← sense, i∈ {0,1,2}
   icv[3]         ← sense
   clock          ← clock+1

1: m[0,i]         ← input[i], i∈ {0,1,2}
   data_out[0]    ← input[1]
   data_out[1]    ← input[0]
   data_out[2]    ← input[0]
   clock          ← clock+1

2: m[1,i]         ← input[i], i∈ {0,1,2}
   data_out[0]    ← m[0,2]
   data_out[1]    ← m[0,2]
   data_out[2]    ← m[0,1]
   clock          ← clock+1

3: m[2,i]         ← input[i], i∈ {0,1,2}
   clock          ← clock+1

4: icv[0]   ← majority(m[0,0], m[1,2], m[2,1])
   icv[1]   ← majority(m[0,1], m[1,0], m[2,2])
   icv[2]   ← majority(m[0,2], m[1,1], m[2,0])
   clock    ← clock+1

5: Actuator       ← filter(icv)
   clock          ← clock+1

6: clock          ← clock+1

7: clock          ← clock+1
```

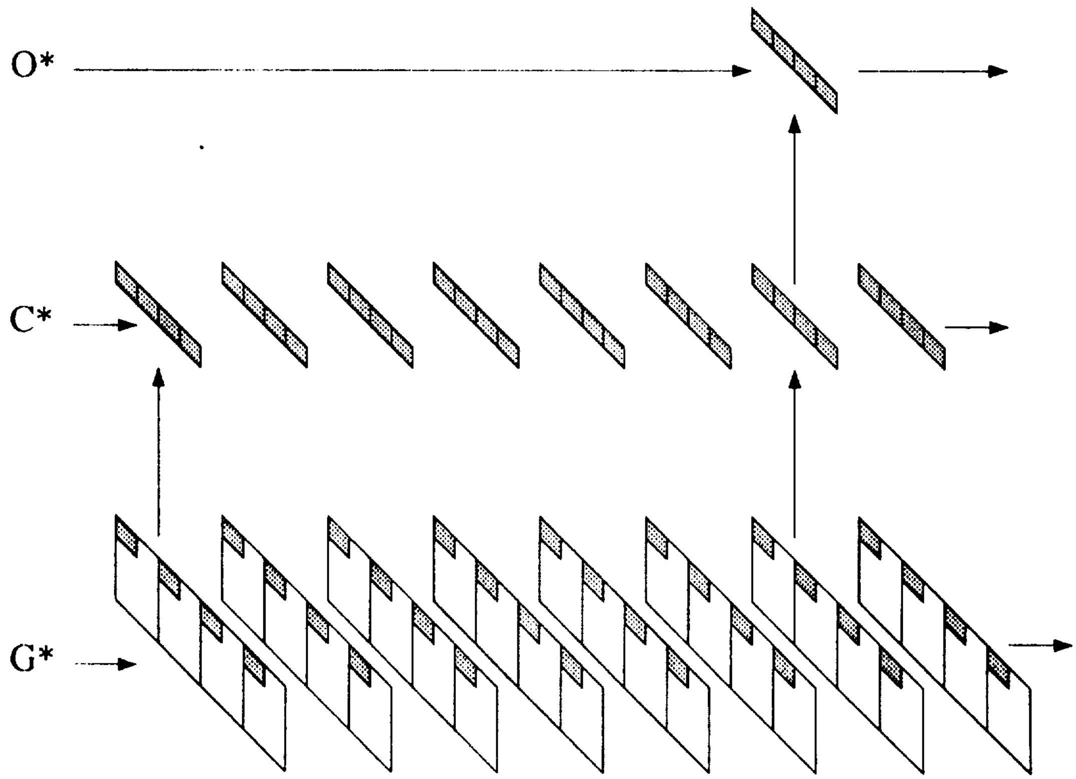
Summary of Device Design

1. Four identical devices.

2. Only internal and external data flow specified, data width not.

3. Filter function constrained to tolerate ICV rotations.

Correctness of Device Design



Device Implementation

by Larry Smith