Abstract—We like to introduce fQuantum, a Quantum Computing Fault Simulator and new quantum computing fault model based on Hadamard, PauliX, PauliY and PauliZ gates, and the traditional stuck-at-1 SA1 and stuck-at-0 SA0 faults. We had close to 100% fault coverage on most circuits. The problem with lower coverage comes from function gates, which we will deal with, in future versions of this paper.

Keywords—fQuantum, Fault Simulator, Quantum Computing

I. INTRODUCTION

QUANTUM computing has been seen as the next stage of computer development. However quantum computer fabrication is prone to errors, just as much as traditional computers. Defect level testing is important to produce high quality quantum computers. We present a new quantum fault model, along side traditional stuck-at fault testing models and we did attain coverage close to 100%.

March 1, 2013

II. QUANTUM FAULT MODEL

The quantum computing fault model is based on concepts similar to classical computing stuck-at models. However there are differences also. First we insert gates instead of setting a line to a particular value. Example gates include Hadamard, PauliX, PauliZ and PauliY gates.

A. Good Machine/Bad Machine Model

Suppose we would like to test a line in the circuit. We first do a good machine simulation. In good machine simulation, the circuit is simulated just the way it is. In bad machine simulation, we take the circuit and insert a quantum unitary transform gate, such as Hadamard, PauliX, PauliY or PauliZ and do a simulation. We then compare good and bad machine outputs, if they dont match, a fault is considered detected otherwise it is declared undetected.

B. Example Circuit

Take the following circuits, a simple Toffoli gate. Refer to figure 1 where there is a fault free circuit and one with fault inserted, i.e. a Hadamard gate. We look at the output spectra of each circuit for an input vector. If they are different a fault is considered detected and fQuantum does reveal fault is detectable.

III. fQUANTUM: FAULT SIMULATOR

fQuantum works by first generating a fault list, doing random test generation RTG, and for each vector in RTG, doing fault simulation and finally reporting coverage.

Algorithm 1 Bad Machine Sim algorithm

if f.gid = 0 then
    InsertInputFault(f)
else
    InitializeRegisters()
    for each gate g in circuit in order do
        if f.gid=g and f is input fault then
            InsertFault(f, g)
        end if
        EvaluateGate(g)
        if f.gid=g and f is output fault then
            InsertFault(f, g)
        end if
    end for
    SetFinalStep()
end if

Algorithm 2 Good Machine Sim algorithm

InitializeRegisters()
for each gate g in circuit in order do
    EvaluateGate(g)
end for
SetFinalStep()
Algorithm 3 FSim algorithm

\begin{verbatim}
GoodMachineSim()
xgReg = xRegister.clone()
ygReg = yRegister.clone()
FaultSimulateBadMachine(f)
 XBReg = xRegister.clone()
 YBReg = yRegister.clone()
if diff(xgReg, XBReg) or diff(ygReg,yBReg) then
  f.detected = Detected
else
  f.detected = Undetected
end if
\end{verbatim}

TABLE I: Fault Coverage

<table>
<thead>
<tr>
<th>Circuit</th>
<th>FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>And.qc</td>
<td>100%</td>
</tr>
<tr>
<td>Deutsch-Josza-1.qc</td>
<td>100%</td>
</tr>
<tr>
<td>H-Fourier.qc</td>
<td>100%</td>
</tr>
<tr>
<td>Shor-437.qc</td>
<td>100%</td>
</tr>
</tbody>
</table>

IV. RESULTS

We report sample coverage on a few test circuits using 10 RTG vectors. Results are shown in table I. We see that we generally did good.

V. CONCLUSION

We built a fQuantum, quantum computing fault simulator and showed it use. In the future, we will, for example, add support for function gates.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[12] Two-state, Reversible, Universal CA in 3D – Miller, Fredkin