Reliability Inversion: A Cautionary Tale

Behrooz Parhami, Life Fellow, IEEE
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560, USA
parhami@ece.ucsb.edu

Abstract

Reliability analysis is often based on worst-case assumptions, so as to produce guaranteed lower-bounds on system survival probability. Reliability engineers try to make the lower-bounds as tight as possible, but sometimes system structure is inherently unfriendly to the derivation of tight bounds. Consider systems A and B, with reliabilities 0.999 and 0.997, respectively. System A is more reliable, but reliability analysis for it may yield a reliability lower-bound of 0.993, whereas the computed lower bound for System B may be 0.995. Because actual reliabilities are unknowable, we have to compare systems based on lower bounds, leading to the declaration of System B as more reliable than System A. I demonstrate that such “reliability inversions” do occur in practice for actual systems under realistic assumptions and point to certain system architectures that are more amenable to producing tight reliability bounds with tractable analytical models or simplified simulation-based models.

Keywords

Dependability; Distributed reconfiguration; Modelability; Reconfigurable arrays; Reliability bound; Reliability modeling; Spare row/column; Switch tracks.

1. Introduction

Reliability inversion is a new concept being introduced in this paper for the first time. Briefly, it leads to a less reliable system being deemed more reliable, owing to uncertainties in reliability modeling. We will define the notion in greater detail in Section 2. Uncertainty in reliability estimates makes the selection of the most-reliable design or system a challenging task, regardless of how the uncertainty is represented: probability, possibility, fuzzy, rough-sets, intervals, and the like [1]. The greater the uncertainties, the harder the comparison. When reliability modeling leads to large uncertainties, we might say that the system isn’t (easily) modelable.

Besides well-known “ilities” (reliability, availability, and other attributes described by words ending in “ility”), dependable system operation is also contingent on lesser-known “ilities” (performability, testability, serviceability, and so on). I propose “modelability” as a new addition to this group of terms. When used qualitatively, the term refers to the ease of accurate reliability modeling. Similar to testability and a number of other “ilities,” which were first introduced as qualitative notions and later quantified, I hope that modelability can some day advance to the quantitative domain.

Modelability is of the same nature as (design for) analyzability, also known as “design for analysis” [2], itself predated by notions such as design for manufacturing (manufacturability). Analyzability requires...
honoring certain design constraints that would allow the use of simpler tools for analysis. In the domain of electronic circuits, design for packageability [3] is quite similar. Both notions constrain the design process, which may seem to lead to higher cost and longer design time. However, somewhat counterintuitively, the end result is often economy and shorter time-to-market.

2. Reliability Inversion Defined

The exact reliability of a system is often unknowable. If we had 100s of identical copies of a system and could run them for decades, observing system failures, we could ascertain the actual reliability with high confidence. The large number of copies and long running times are needed because, at typically high system reliabilities, failures are extremely rare, so to obtain statistically valid results, extensive data collection is required. An alternative is to make simple, pessimistic assumptions about subsystems and their interactions, in an analytic or simulation model, to derive a lower bound on reliability. Models don’t completely eliminate the need for experimentation, as model parameters may be derived, and models themselves tuned, based on experimental observations.

The actual system reliability could be much better than a model-based lower bound. We see in Fig. 1 that even though System A is more reliable than System B (if we somehow knew the actual reliabilities), the model-based lower bounds ascribe a higher reliability to System B. We thus have no choice but to recommend System B over System A as being more reliable. This situation is what I call “reliability inversion,” in analogy to the similarly disruptive phenomenon of “priority inversion” in real-time task scheduling [4] that wreaked havoc during the Mars Pathfinder mission of the late 1990s [5].

Reliability is of course a function of time. Generally, one can’t say that a system is always more reliable than another one. One system may be more reliable for short mission times, while another fares better for long mission durations. So, let’s enter the time factor into the notion of reliability inversion. The actual and modeled reliabilities of Systems A and B are depicted in Fig. 2. With regard to unknowable actual reliabilities, System A is better for short mission times, whereas System B does better over the long run. With regard to model-based bounds, however, System B is uniformly better and would be the preferred choice in all cases.

We may call a system for which the guaranteed lower bound is very close to actual reliability a highly modelable system. Conversely, a system has poor modelability when the bound is much lower than the actual reliability. Of the two systems depicted in Fig. 2, System B has better modelability than system A, although its actual reliability is worse for short mission times. If we were to choose System A or B for a particular critical application, we would choose B, because we have no way of knowing the true reliabilities. All we have to go by are the bounds provided by reliability models, and the bound for System B is uniformly better than that of A.

It may be argued that reliability inversion is a blessing in disguise. Because models are imperfect, in the sense of not taking all failure causes and mechanisms into account, perhaps the wider gap between the lower bound and the actual reliability can provide a safety margin to guard against unpredictable or overlooked failure causes and mechanisms. However, best practices in reliable system design and tenets of safety engineering require us to provide deliberate and predictable safety margins, rather than rely on a margin materializing by happenstance. While it is true that playing too close to the edge may be dangerous, especially in highly complex systems [6], we prefer to distance ourselves from the edge deliberately, rather than haphazardly.
3. Reconfigurable Processor Arrays

In this section, I introduce a class of reconfigurable processor arrays for use as examples to demonstrate reliability inversion. In particular, a special case of redundancy and reconfiguration in which an $n \times n$ mesh or grid of processing elements (PEs) is augmented with one spare row and one spare column, for a redundancy ratio of $(2n + 1)/n^2 = O(1/n)$, along with embedded switches that allow processors to change their row or column neighbors when nodes malfunction. This constitutes a good example to pursue, in view of its extensive assessment and documentation [7] [8].

In the references just cited, and the examples we will draw upon, reconfiguration is performed to return a processor array with malfunctioning nodes to its initial healthy configuration, so as to be able to execute the original $n \times n$ mesh algorithms without modification. Specifically, we are not considering the kind of reconfiguration that extends the computational power of the array (in a complexity-theory sense), allowing it to achieve significant speed-up in performing certain computations via dynamic adaptation [9].

To make the examples even more concrete, we will consider a $5 \times 5$ guest array within a $6 \times 6$ host array, that is, one with a spare row (at the bottom) and a spare column (on the right), as depicted in Fig. 3. Originally, the nodes in the topmost $5$ rows and the leftmost $5$ columns are active, with the configuration changing as nodes malfunction. When a PE becomes unusable, it is dealt with in various ways. It can be bypassed in its respective row or column, and/or it can be configured out by downward shifting of the rows or rightward shifting of the columns (Fig. 4).

We won’t discuss the details of the switching mechanisms and algorithms that effect reconfiguration (see, e.g., [10]), mentioning only that any double-PE malfunction can be tolerated through reconfiguration, but there are worst-case patterns of $3$ unusable PEs that exceed the scheme’s reconfigurability [11]. As seen in Fig. 3, we have $60$ switches, arranged on tracks between PE rows/columns, to allow salvaging a $5 \times 5$ guest array from a $6 \times 6$ host. More generally, given an $n \times n$ original array embedded in an $(n + 1) \times (n + 1)$ augmented array, the number of switches required is $2n(n + 1)$, that is, linear in the number of PEs.

4. Centralized vs. Distributed Switching

To demonstrate that reliability inversion isn’t just a theoretical curiosity, we show that it can occur in actual systems under realistic conditions. We consider the reconfiguration scheme depicted in Figs. 3 and 4 as an example, focusing on a $5 \times 5$ guest network embedded in a $6 \times 6$ host array. The system remains functional, after reconfiguration, if all the switches work and if $34$ of the $36$ PEs are functional. Let the PE failure rate be $\lambda$ and the switch failure rate be $\sigma$. Then,

\[
\begin{align*}
\text{Module/PE reliability} & = r = e^{-\lambda t} \quad (1) \\
\text{Overall switching reliability} & = e^{-(60\sigma)t} \quad (2) \\
\text{System reliability} & = e^{-(60\sigma)t}R_{34\text{-out-of-36}}(r) \quad (3)
\end{align*}
\]

where $R_{k\text{-out-of-}n}(r)$ is the $k$-out-of-$n$ reliability for modules of uniform reliability $r$. Computationally:

\[
\begin{align*}
R_{34\text{-out-of-36}}(r) & = r^{36} + 36r^{35}(1-r) + (36\times35/2)r^{34}(1-r)^2 \\
& = r^{34}[r^2 + 36r(1-r) + 630(1-r)^3] \\
& = r^{34}[595r^2 - 1224r + 630] \\
& = r^{34}[1 + (1-r)(629 - 595r)] \quad (4)
\end{align*}
\]

Substituting Eq. (4) into Eq. (3) and using $\sigma = 0.01\lambda$, we get the reliability plot shown as a gray line in Fig. 5.
We next consider a reconfiguration scheme based on the use of multiplexers (muxes) within PEs, so that each PE can select its north/above and west/left neighbors from among three possibilities, as shown in the right-hand panel of Fig. 6. Again, we delete some details that demonstrate the equivalence of the two schemes with regard to reconfigurability. Now, each PE becomes a tad more complex, increasing its failure rate to $\lambda + \alpha \sigma$, where $\sigma$ is the failure rate of the original track switches and $\alpha$ is the distribution overhead, representing the increase in switch hardware complexity as a result of the distribution process. We now have the following reliability equations:

Module/PE reliability = $r' = e^{-(\lambda + \alpha \sigma)t}$

System reliability = $R_{34-out-of-36}(r')$  \hspace{1cm} (5)

In our numerical example, we take $\alpha = 2$ as a reasonable pessimistic value, given the presence of 60/36 $\equiv$ 1.67 switches per PE in the centralized scheme, with a $2 \times 2$ switch built from two 2-to-1 muxes. The distributed scheme needs two 3-input muxes per PE.

The resulting unreliability curve is shown as the heavy black line in Fig. 5. We note that the reliability advantage of the distributed scheme declines as $\lambda t$ increases. This is because for large $\lambda t$ values, PE malfunctions will dominate, making switching differences less relevant. If we extend the curves for even larger values of $\lambda t$, up to 1, say, unreliabilities will approach 1, rendering the systems both indistinguishable and practically useless.

5. Demonstrating Reliability Inversion

We see that for $\lambda t$ values in the range of practical interest, distributed switching offers uniformly higher reliability lower bound. Of course, as noted earlier, this does not mean that the reliability of the distributed scheme is always higher, only that it lends itself to the derivation of tighter bounds. To complete our demonstration of potential reliability inversion in a practical setting, we need to show that under some reasonable assumptions, the centralized system may in fact have higher reliability, despite its poorer reliability lower bound.

Consider modeling the centralized switches in greater detail, rather than lumping all switching hardware together into a hard core modeled by eq. (2). A lot of extra work would be required in this case, as switch failures and failure interactions depend on both the switch architecture and implementation technology. Let us assume an implementation technology for which a switch can be assumed not to fail unless we attempt to change its state. If, on average, only 6 operational switches are needed for correct reconfiguration (with high probability, reconfiguration entails bypassing a single PE), then the pertinent system reliability equation is:

System reliability = $e^{-(6\sigma)t}R_{34-out-of-36}(r')$  \hspace{1cm} (7)

Equation (7) does not yield a reliability lower bound, so it cannot be used for system comparisons with certifiable outcome. However, it suffices for the purpose of demonstrating that centralized switching can have higher reliability than the distributed scheme under certain conditions. A plot of Eq. (7) is shown as the dotted gray line in Fig. 5. We see that the dotted line (possibly) goes below the heavy black line beginning at $\lambda t = 10^{-2}$. We can verify that this is indeed the case by looking a few data points based on Eq. (3) and Eq. (7) (Table 1).

<table>
<thead>
<tr>
<th>$\lambda t$</th>
<th>Eq. (3)</th>
<th>Eq. (7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td>0.994</td>
<td>0.994</td>
</tr>
<tr>
<td>0.020</td>
<td>0.964</td>
<td>0.965</td>
</tr>
<tr>
<td>0.050</td>
<td>0.735</td>
<td>0.742</td>
</tr>
<tr>
<td>0.100</td>
<td>0.308</td>
<td>0.319</td>
</tr>
</tbody>
</table>

Table 1: Reliability inversion data points.
To provide an intuitive feel for our conclusions, we note that the reliability bound for centralized reconfiguration isn’t tight because we had to proceed with the highly pessimistic assumption that the entire switching network forms a critical core. We had no choice here, as which switches will need to be re-programmed for a particular pattern of PE malfunctions is unknown. In the distributed scheme, on the other hand, switches are integrated into the PEs, thus as long as 34 of the 36 PE-switch modules are functional, we can successfully reconfigure the system. We do not care about the health of the switching mechanism any more than we care about PE health. The system has no single point of failure.

Even though we considered only a relatively small example, the difference between reliabilities of the centralized and distributed schemes only grows as we enlarge the array. So, the results do scale up to very large PE arrays of practical interest. As mentioned previously, larger arrays will show greater benefits for distributed reconfiguration in terms of the differences between the lower bounds. They will also amplify the fairly small inversion appearing in Table 1.

6. Conclusion

In this paper, I have tried to raise awareness of the notions of “reliability inversion” and “modelability,” using a concrete example for experimental validation of the abstract ideas. Even though work on reconfiguration schemes and algorithms for degradable processor arrays has continued unabated since the early work previously cited [12], [13], [14], [15], such variations, extensions, and improvements do not affect the formulation of the reliability inversion concept. Design and reliability modeling considerations for reconfigurable 2D processor arrays with centralized and distributed switching will be taken up in a companion paper [16].

Besides modelability benefits, distributed reconfiguration of 2D processor arrays also leads to a more regular and modular design, hence enjoying greater packageability as well as suitability for VLSI realization. This is an important side benefit that is similar to benefits cited for design for analyzability [2]. The perils discussed here in connection with reliability inversion can be summed up in the following maxim: A benefit that is not observable to us, because models don’t show it, is no benefit at all.

References