

06.20

Computer

CYBERTHREATS



IEEE



IEEE
COMPUTER
SOCIETY

vol. 53 no. 6

www.computer.org/computer



Reliability Inversion: A Cautionary Tale

Behrooz Parhami, University of California, Santa Barbara

Reliability analysis is often based on worst-case assumptions to produce guaranteed lower bounds on system survival probability. Reliability engineers make lower bounds as tight as possible, but sometimes system structure is unfriendly to the derivation of tight bounds. Unfortunately, loose reliability lower bounds make it difficult to compare design alternatives or to select among competing systems.

Reliability inversion is a new concept being introduced in this article for the first time. Briefly, it leads to a less reliable system being deemed more reliable because of uncertainties in reliability modeling. We will define the idea in greater detail in the next section. 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 is not (easily) modelable.

Besides well-known “ilities” (reliability, availability, and other attributes described by words ending in

Digital Object Identifier 10.1109/MC.2019.2958907
Date of current version: 4 June 2020

“ility”), dependable system operation is also contingent on lesser known “ilities” (performability, testability, serviceability, and so on). We 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, we hope that modelability can someday 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 concepts such as design for manufacturing (manufacturability). Analyzability requires honoring certain design constraints that allow the use of simpler tools for analysis. In the domain of electronic circuits, design for packageability³ is quite similar. Both notions constrain the design process, which may seem to lead to higher costs and longer design times. However, somewhat counterintuitively, the end result is often economy and shorter time to market.

RELIABILITY INVERSION DEFINED

The exact reliability of a system is often unknowable. If we had hundreds of identical copies of a system and could run them for decades, observing system failures, we could ascertain the actual reliability with high confidence. A large number of copies and long running times would be 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 do not 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 Figure 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 we call *reliability inversion*, in analogy to the similarly disruptive phenomenon of priority inversion in real-time task scheduling⁴ that wreaked havoc during the Mars Pathfinder mission of the late 1990s.⁵

Reliability is, of course, a function of time. Generally, one cannot 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 us enter the time factor into the notion of reliability inversion. The actual and modeled reliabilities of Systems A and B are depicted in Figure 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

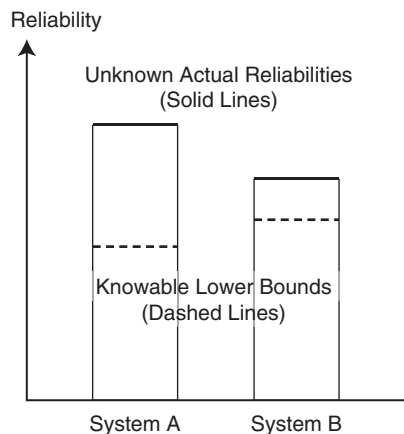


FIGURE 1. Reliability inversion.

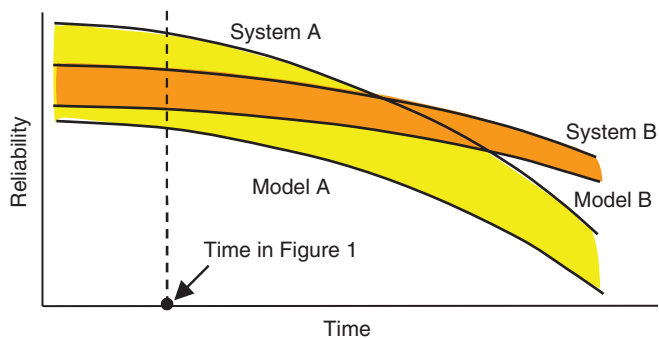


FIGURE 2. True reliability versus modeled lower bound.

lower than the actual reliability. Of the two systems depicted in Figure 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.

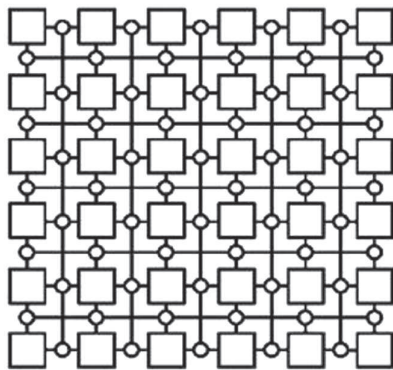


FIGURE 3. A square 5 × 5 array of PEs with a spare row (bottom) and a spare column (right).

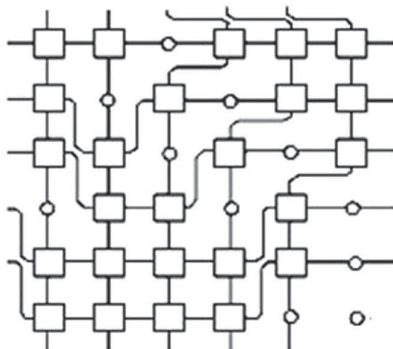


FIGURE 4. The array of Figure 1, configured to salvage a 5 × 5 healthy array from a 6 × 6 injured one.

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.

RECONFIGURABLE PROCESSOR ARRAYS

In this section, we 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 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 speedup in performing certain computations via dynamic adaptation.⁹

To make the examples even more concrete, we will consider a 5×5 guest array within a 6×6 host array, that is, one with a spare row (at the bottom) and a spare column (on the right), as depicted in Figure 3. Originally, the nodes in the topmost five rows and the leftmost five columns are active, with the configuration changing as nodes malfunction. When a PE becomes unusable, it can be 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 the rows or rightward shifting the columns (Figure 4).

We will not discuss the details of the switching mechanisms and algorithms that affect reconfiguration,¹⁰ mentioning only that any double-PE malfunction can be tolerated through reconfiguration, but there are worst-case patterns of three unusable PEs that exceed the scheme’s reconfigurability.¹¹ As can be seen in Figure 3, we have 60 switches, arranged on tracks between PE rows/columns, to allow salvaging a 5×5 guest array from a 6×6 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.

CENTRALIZED VERSUS DISTRIBUTED SWITCHING

To demonstrate that reliability inversion is not just a theoretical curiosity, we show that it can occur in actual systems under realistic conditions. We consider the reconfiguration scheme

depicted in Figures 3 and 4 as an example, focusing on a 5×5 guest network embedded in a 6×6 host array. The system remains functional after reconfiguration if all of the switches work and if 34 of the 36 PEs are functional. Let the PE failure rate be λ and the switch failure rate be σ . Then,

$$\text{Module/PE reliability} = r = e^{-\lambda t} \quad (1)$$

$$\text{Overall switching reliability} = e^{-(60\sigma)t} \quad (2)$$

$$\begin{aligned} \text{System reliability} \\ = e^{-(60\sigma)t} R_{34\text{-out-of-36}}(r), \end{aligned} \quad (3)$$

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

$$\begin{aligned} R_{34\text{-out-of-36}}(r) &= r^{36} + 36r^{35}(1-r) \\ &\quad + (36 \times 35 / 2)r^{34}(1-r)^2 \\ &= r^{34}[r^2 + 36r(1-r) \\ &\quad + 630(1-r)^2] \\ &= r^{34}[595r^2 - 1224r \\ &\quad + 630] \\ &= r^{34}[1 + (1-r)(629 \\ &\quad - 595r)]. \end{aligned} \quad (4)$$

Substituting (4) into (3) and using $\sigma = 0.01\lambda$, we get the reliability plot shown as a gray line in Figure 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 Figure 6(b). 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 σ is the failure

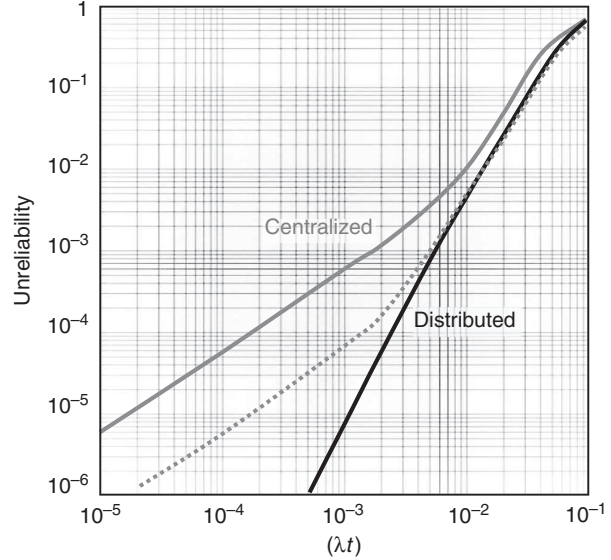


FIGURE 5. System unreliability for a reconfigurable array of PEs as a function of λt for one PE.

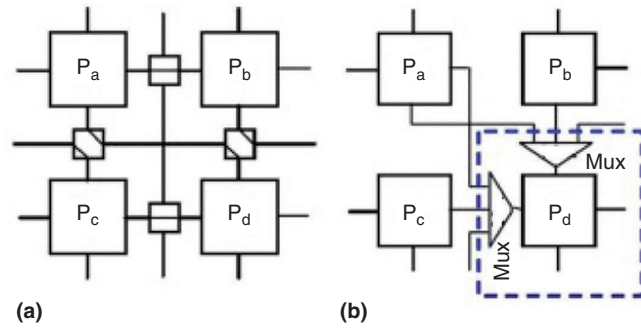


FIGURE 6. (a) External (centralized) switches can be replaced by (b) muxes within PEs (distributed).

rate of the original track switches and α 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:

$$\text{Module/PE reliability} = r' = e^{-(\lambda + \alpha\sigma)t} \quad (5)$$

$$\text{System reliability} = R_{34\text{-out-of-36}}(r'). \quad (6)$$

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

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

DEMONSTRATING RELIABILITY INVERSION

We see that for λt 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 (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 six operational switches are needed for correct reconfiguration (with high probability, reconfiguration entails bypassing

a single PE), then the pertinent system reliability equation is

$$\text{System reliability} = e^{-(6\sigma)t} \times R_{34\text{-out-of-36}}(r). \quad (7)$$

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 (7) is shown as the dotted gray line in Figure 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 (3) and (7) (Table 1).

To provide an intuitive feel for our conclusions, we note that the reliability bound for centralized reconfiguration is not 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 reprogrammed 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.

In this article, we 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 papers previously cited,¹²⁻¹⁵ such variations, extensions, and improvements do not affect the formulation of reliability inversion. Design and reliability modeling considerations for reconfigurable 2D processor arrays with centralized and distributed switching will be taken up in a companion article.¹⁶

Besides modelability benefits, distributed reconfiguration of 2D processor arrays also leads to a more regular and modular design, hence providing greater packageability as well as suitability for realization as very large scale integrated circuits. This is an important side benefit that is similar to those cited for design for analyzability.²

The perils discussed here in connection with reliability inversion can be summed up by the following maxim: A benefit that is not observable to us,

TABLE 1. Reliability inversion data points.				
λt	0.010	0.020	0.050	0.100
Equation (3)	0.994	0.964	0.735	0.308
Equation (7)	0.994	0.965	0.742	0.319

because models don't show it, is no benefit at all. 

REFERENCES

1. E. Zio and N. Pedroni, "Methods for representing uncertainty: A literature review," French Foundation for an Industrial Safety Culture, Toulouse, Mar. 2013. Accessed on: Apr. 15, 2020. [Online]. Available: <https://www.foncsi.org/fr/publications/cahiers-securite-industrielle/literature-review-uncertainty-representation/CSI-uncertainty-representation.pdf>
2. R. Suri and M. Shimizu, "Design for analysis: A new strategy to improve the design process," *Res. Eng. Design*, vol. 1, no. 2, pp. 105–120, 1989. doi: 10.1007/BF01580204.
3. P. H. Dehkordi and D. W. Bouldin, "Design for packageability—Early consideration of packaging from a VLSI designer's viewpoint," *Computer* vol. 26, no. 4, pp. 76–81, Apr. 1993. doi: 10.1109/2.206519.
4. D. Locke, L. Sha, R. Rajikumar, J. Lehoczky, and G. Burns, "Priority inversion and its control: An experimental investigation," *ACM SIGADA Ada Lett.*, vol. VIII, no. 7, pp. 39–42, 1988. doi: 10.1145/59368.59374.
5. G. Reeves, "What really happened on Mars," *Risks Dig.*, vol. 19, no. 54, 1998. Accessed on: Apr. 15, 2020. [Online]. Available: https://www.cs.unc.edu/~anderson/teach/comp790/papers/mars_pathfinder_long_version.html
6. S. Dekker, *Drift Into Failure: From Hunting Broken Components to Understanding Complex Systems*. Boca Raton, FL: CRC, 2011.
7. M. Chean and J. A. B. Fortes, "A taxonomy of reconfiguration techniques for fault-tolerant processor arrays," *Computer* vol. 23,

ABOUT THE AUTHOR

BEHROOZ PARHAMI is a professor of electrical and computer engineering at the University of California, Santa Barbara. His research interests include computer arithmetic, parallel processing, and dependable computing. He has served on the editorial boards of several journals, including *IEEE Transactions on Computers* and *IEEE Transactions on Sustainable Computing*. He is a Life Fellow of the IEEE. Contact him at parhami@ece.ucsb.edu.

- no. 1, pp. 55–69, Jan. 1990. doi: 10.1109/2.48799.
8. M. Sami and R. Stefanelli, "Reconfigurable architectures for VLSI processing arrays," *Proc. IEEE*, vol. 74, no. 5, pp. 712–722, 1986. doi: 10.1109/PROC.1986.13533.
9. Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster, "The power of reconfiguration," in *Proc. Int. Colloquium Automata, Languages, and Programming*, 1991, pp. 139–150. doi: 10.1007/3-540-54233-7_130.
10. M. Fukushi, and S. Horiguchi, "Reconfiguration algorithm for degradable processor arrays based on row and column rerouting," in *Proc. 19th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, 2004, pp. 496–504. doi: 10.1109/DFTVS.2004.1347875.
11. B. Parhami, "Dependable computing: A multilevel approach," Nov. 19, 2019. Accessed on: Apr. 15, 2020. [Online]. Available: https://www.ece.ucsb.edu/~parhami/text_dep_comp.htm
12. G. Jiang, J. Wu, and J. J. Sun, "Efficient reconfiguration algorithms for communication-aware three-dimensional processor arrays," *Parallel Comput.*, vol. 39, no. 9, pp. 490–503, 2013. doi: 10.1016/j.parco.2013.04.005.
13. J. Wu, T. Srikanthan, G. Jiang, and K. K. Wang, "Constructing sub-arrays with short interconnects from degradable VLSI arrays," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 4, pp. 929–938, 2013. doi: 10.1109/TPDS.2013.114.
14. J. Wu, N. Liu, S.-K. Lam, and G. Jiang, "Shortest partial path first algorithm for reconfigurable processor array with faults," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 1198–1203. doi: 10.1109/TrustCom.2016.0194.
15. J. Qian, W. Cao, J. Hu, J. Zhang, Z. Xu, and Z. Z. Zhou, "Satisfiability-based method for reconfiguring power efficient VLSI array," *IEICE Electron. Express*, vol. 13, no. 23, pp. 1–11, 2016. doi: 10.1587/elex.13.20160930.
16. B. Parhami, "Reliability and modularity advantages of distributed switching for reconfigurable 2D processor arrays," unpublished.

IEEE COMPUTER SOCIETY
DIGITAL LIBRARY
Access all your IEEE Computer Society subscriptions at
computer.org
/mysubscriptions