RBD Tools Using Compression, Decompression, Hybrid Techniques to Code, Decode, and Compute Reliability in Simple and Complex Embedded Systems

Mehmet Sahinoglu, *Senior Member, IEEE,* and Chittoor V. Ramamoorthy, *Life Fellow, IEEE*

***Abstract*—A large amount of work is in progress on reliability block diagramming (RBD) techniques. Another body of dynamic research is in digital testing of embedded systems with very large scale integration (VLSI) circuits. Each embedded system, whether simple or complex, can be decomposed to consist of components (blocks) and interconnections or transmissions (links) within an s-source (input) and t-target (output) setup. There will be three tools proposed in this study. The first tool, using a novel "compression algorithm" is capable of reducing any complicated series-parallel system (not complex) to a visibly easy sequence of series and parallel blocks in a reliability block diagram by first finding all existing paths, then algorithmically compressing all redundant component duplications, and finally calculating an exact reliability and creating an encoding of the topology. A second tool is to decode and retrieve an already coded $s-t$ dependency relationship using post-fix notation for series-parallel or complex systems. A third tool is an approximate fast upper-bound (FUB) $s-t$ reliability computing algorithm designed for series–parallel systems, to perform state enumeration in a *hybrid* form assisted by the Polish encoding approach on non-series-parallel complex systems to compute the exact s (source)—t (target) reliability. Various examples illustrate how these tools work satisfactorily in unison. Further research with the OVERLAP method is in progress to reduce the computation speed by a thousand fold for a grid of 19 nodes without sacrificing any accuracy.**

***Index Terms*—Code–decode, complex, compression, hybrid, reliability block diagramming (RBD), series-parallel, $s-t$ reliability.**

I. Introduction and Motivation

RELIABILITY block diagramming (RBD) has been an active area of research for decades, even more so now with the advent of the embedded systems [1]–[11]. This paper explores to describe and compute the $s-t$ reliability in such (embedded) systems through an RBD approach. It is assumed that the input data required, such as reliability or availability including the aspect of security for each component and link in the RBD approach, is correctly facilitated by improving the very large scale integration (VLSI) testing techniques [24]–[30]. Earlier, simple or complicated series–parallel systems are studied to demonstrate that these networks can be encoded using a modified Polish notation employing postfixes [12], [17], [19]–[22]. The "compression" algorithm through a user-friendly and graphical Java application computes the reliability of any series–parallel network, no matter how large or complicated it is. Furthermore, the encoded topology can be transmitted remotely and then reverse-coded to reconstruct the original network diagram for purposes of securing classified information and saving space, a project which is also in progress nearing completion.

Interest in considering reliability during design of computer communications networks with a large number of nodes and connecting links, such as those found in hospitals, universities, electricity distribution, gas pipelines, military, or internet has increased in recent years. Due to geographical and physical constraints in such critical systems, designers at the initial or improvement stages usually base their decisions on approximate or upper-bound estimates of reliability to compute a given ingress (source) to egress (target) reliability. This practice may be deceptive, erroneous and overly optimistic due to computational complexity when reliability remains of a crucial importance that means human life and health.

The graphical screening ease and convenience of this algorithm are advantageous for planners and designers trying to improve system reliability by allowing a quick and efficient intervention that may be required at a dispatch center to observe routine operations and/or identify solution alternatives in case of a crisis.

The Boolean decomposition and binary enumeration algorithms or BDD [13]–[16] are outside the scope of this work, although it illustrates a new hybrid solution with the Polish notation. The proposed algorithm, through a user-friendly and graphical Java applet, computes the reliability of any complex series-parallel network. Furthermore, the coded topology can be transmitted remotely and then reverse-engineered to reconstruct the original network diagram for purposes of securing classified information and saving space.

All current exact computational algorithms for general networks are based on enumeration of states, minpaths, or mincuts [2], [3]. Network reliability estimation has been used successfully for nontrivial-sized networks using neural networks and heuristic algorithms in [7] and [8] as well as employing a "concurrent error detection" approach by the coauthor of this research as in [18]. Other researchers have used efficient Monte

Manuscript received August 12, 2004; revised July 8, 2005.

M. Sahinoglu is with the Department of Computer Sciences, Troy University, Montgomery, AL 36103 USA (e-mail: mesa@troy.edu).

C. V. Ramamoorthy is with the Department of Electrical Engineering and Computer Sciences, Computer Science Division, University of California, Berkeley, CA 94720 USA (e-mail: ram@csce.berkeley.edu).

Digital Object Identifier 10.1109/TIM.2005.855103

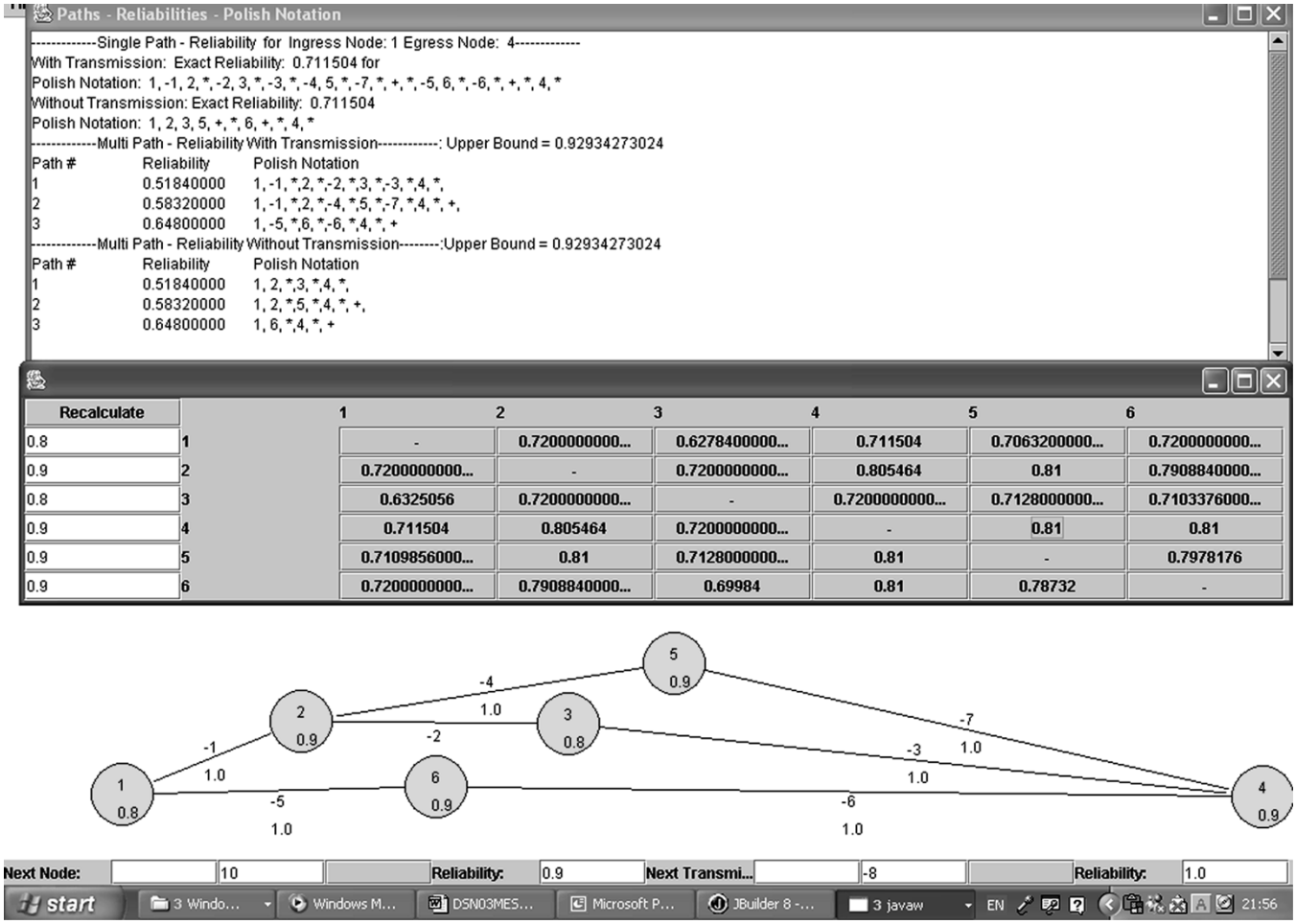


Fig. 1. Screen 1: RBD and matrix of $s-t$ reliabilities from text in [11], Table I: reliability results of $(s=1, t=4)$.

Carlo simulation [4], [5]. Bounds like Jan's upper-bound, to reduce the complexity of computations, are approximate [3]. A thorough analysis is given by Colbourn [1].

II. Illustrative Simple Example

For this paper, some series–parallel examples will be used to illustrate the algorithm. The RBD tool is a Java application. The example with an exact reliability solution is simple and tractable by hand. The method used in this paper, exact reliability block diagram calculation (ERBDC), is an exact calculation of $s-t$ reliability but tractable for large networks. As an example of this method, a Java applet (Fig. 1, Table I) examines Example 6.3 of Fig. 6.4 of [11]. The node failures are all $q=0.1$, except for $q_1=q_3=2q=0.2$. (Note, links have zero failure probability for simplicity.) Let T denote a tie-set. If $q=$ failure probability for all components, then $\Pr(\mathrm{T_1 U\ T_2 U\ T_3}) = \Pr(164)+\Pr(1234)+\Pr(1254)-\Pr(12364)-\Pr(12654)-\Pr(12354)+\Pr(123654) = (1-2q)(1-2q+q^2)(1+q-2q^3) = 0.8*0.81*1.098 = 0.711504$, which can be observed in the array 1, 4 of Fig. 1, Table I. The ingress–egress relationship is also tabulated in Fig. 1, Table I by Polish, or postfix notation where postfixes $*$ and $+$ denote two-at-a-time series and parallel components, respectively, by Sahinoglu and Libby [6]. The upper-bound of system reliability is calculated by treating the three paths in parallel, Reliability $= 1-(1-.0648)(1-.5184)(1-.5832) = 0.92934$, as shown in Fig. 1, Table I.

However, when the number of components is increased to many more, it becomes tedious to arrange the network in a nice sequence of series and parallel subsystems. The "compression algorithm" does that and also calculates the exact $s-t$ reliability for simple series–parallel (not complex) but complicated looking systems, as it will be shown in Section III.

III. Compression Algorithm and Various Applications

The algorithm to facilitate a simpler way to compute the $s-t$ reliability is as follows. In a parallel set composed of $i, \mathrm{j}, k, l, m, \ldots$ paths; at each item i, compress for each following item j. If i can combine with j; then do so, and remove j. If not, then keep it and then compress again with the next kth path, until all of the parallel paths have been exhausted. At the end, there is a single compressed path RBD from ingress to egress node. A line between two nodes is treated as a series component between the two, such as in Fig. 1, Table I. Line-1 connecting the nodes 1 and 2 is expressed as in 1, $-1, *$, 2, $*$. Two components in parallel are designated as 1, 2, $+$ [6],

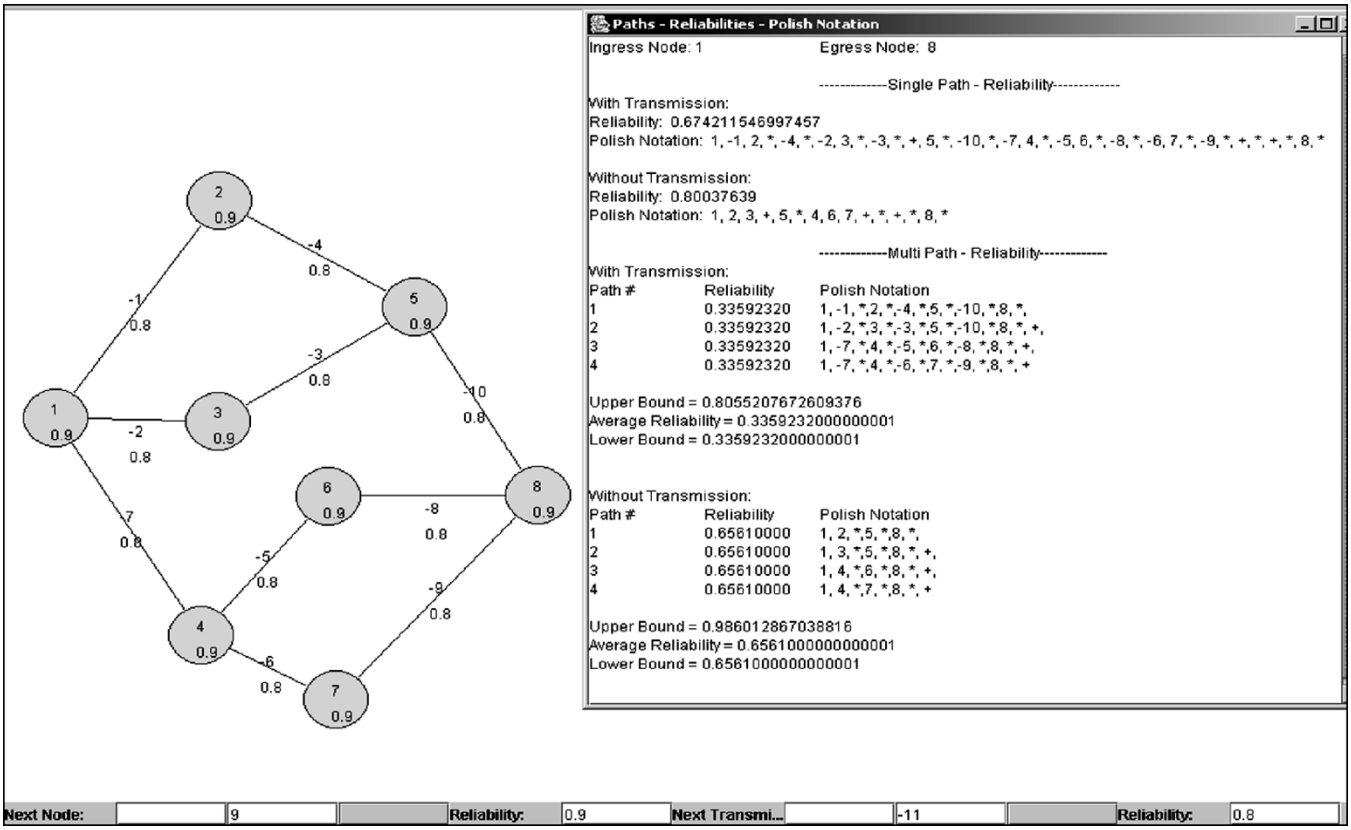


Fig. 2. Screen 2: $s-t$ reliability $(\mathrm{s}=1, \mathrm{t}=8)$ for a series-parallel network. RBD. Table II: Results.

[12], [17], [19]–[22]. Let us take the following series–parallel example as in Fig. 2, Table II.

The + postfixes at the end of each serial path denote that those paths will be combined in parallel to calculate the upper bound. Otherwise, each path has all of its components connected in series denoted by a succession of $*$ postfixes. No more than two consecutive components are allowed for each postfix. One will take each path in ascending or descending sequence as convenient to compare and contrast. The compression is executed as many times as existing paths. See Table III for an illustration. A complicated and not obviously tractable series–parallel Ding–Dong1 network of Fig.3 (Screens 3 and 4) will also be studied.

On the other hand, Fig. 3 depicts a simulated LAN operation, consisting of 22 links and 19 nodes. This network has all nodes with a reliability of 0.9 and links with a reliability of 0.8, respectively. Note that the lines are assigned negative prefixes, and $s=1$ and $t=13$ are the ingress (source) and egress (target) nodes, respectively. The network can be translated into a Polish (dependency) notation as in Table IV to calculate the $s-t$ reliability. The algorithm offers a user-friendly graphical interface, speed, and accuracy especially in the event of imperfect links beyond a secure environment to transport on the net through a reverse engineering process, given in the final section of this paper.

Another application for the compression algorithm for an 11-node simple series–parallel network is as follows. The $s-t$ reliability analysis using the new Polish-notation approach is presented further.

Considering path reliability, the paths or tie-sets are as follows when all links are assumed to operate with full reliability of unity.

Without Link:

Path#	Reliability	Polish Notation
1	0.72900000	$5,1,*,9,*,$
2	0.65610000	$5,1,*,10,*,9,*,+,$
3	0.47829690	$5,2,*,3,*,4,*,11,*,1,*,9,*,+,$
4	0.43046721	$5,2,*,3,*,4,*,11,*,1,*,10,*,9,*,+$

The $+$ signs at the end of each serial path denote that those paths will be combined in parallel to calculate the upper bound. Otherwise, each path has all its components connected in series denoted by a succession of $*$ postfixes. One will take each path in ascending or descending sequence as convenient to compare and contrast as follows:

5, 1, *,9, *
5, 1, *,10, *,9, *
5 _____**1**_____**9**, to merge with
|-----10--------|

5, 2, *,3, *,4, *,11, *,1, *,9, * to converge to:

5 _____**1** _____**9**, which will finally merge with
|----------10--------|
|2----3-------11-----------|

5, 2, *,3, *,4, *,11, *,1, *,10, *,9, * to converge to:

5 _____**1** _____**9**
| |----------10-----|
|2--- 3-------11---------- |
|--4----|

Since the straight line (with a reliability of unity as assumed) between nodes 5–1 and 1–9 dominates, the rest of the branches

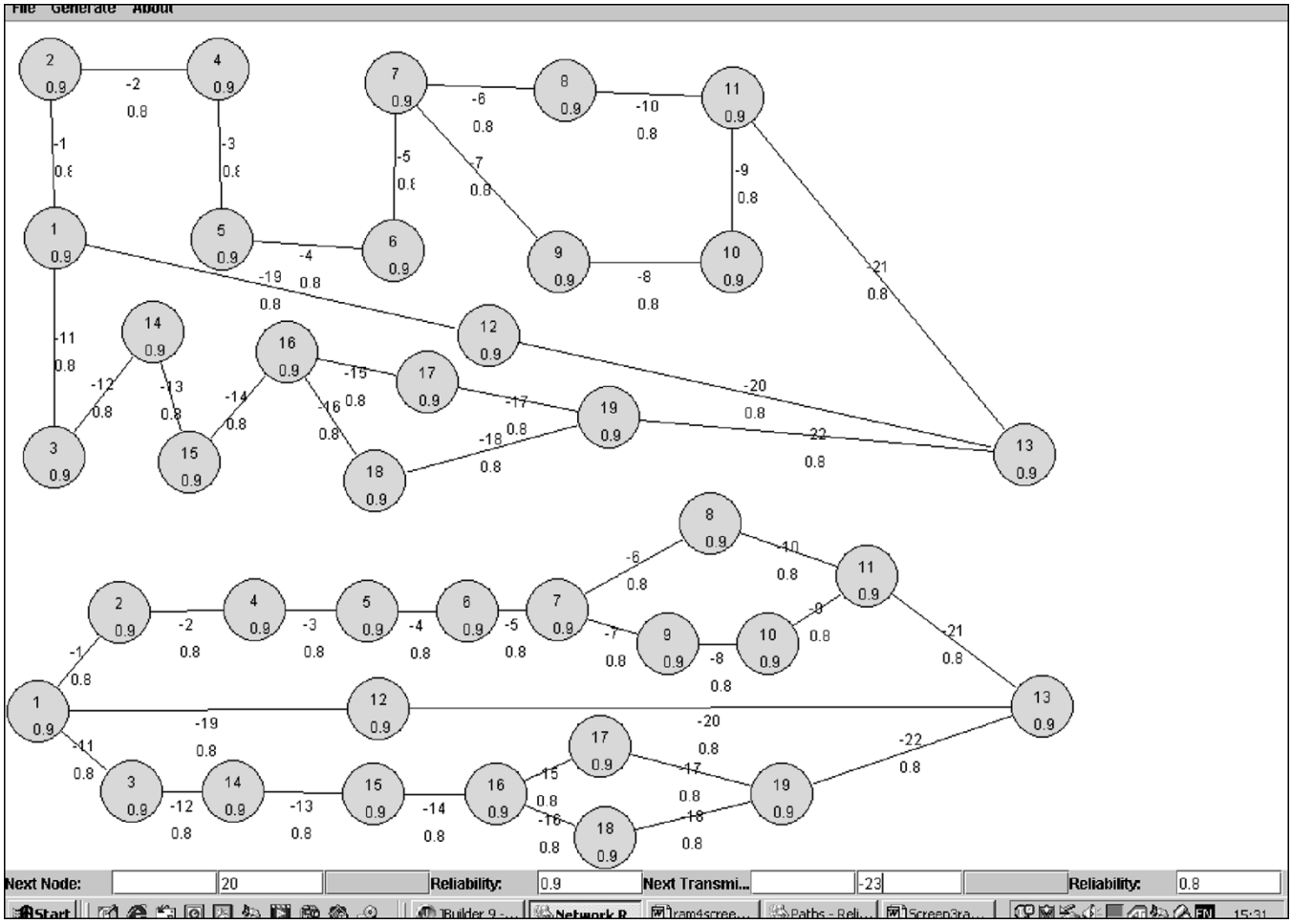


Fig. 3. Screens 3 and 4: Topology of Fig. 3 for Ding–Dong1 network as reduced to an easier Fig. 4.

are ineffective. Therefore, the system reliability is the series connection of three nodes 5, 1, and 9; i.e., 5, 1, $*$, 9, $*$ using Polish notation. That is, $0.9^3 = 0.729$. If links have nonunity reliability, then links have to be multiplied in series as well.

When the links have failures operating with nonunity reliability, then it is a different scenario. The following table is utilized to compose the finalized single-path RBD to calculate the exact reliability. Note, the negative digits denote links and sole digits show the nodes. See Fig. 4 (Screen 5).

With Link:

Path#	Reliability	Polish Notation
1	0.72900000	$5,-1,*,1,*,-11,*,9,*,$
2	0.65610000	$5,-1,*,1,*,-13,*,10,*,-12,*,9,*,+,$
3	0.47829690	$5,-2,*,2,*,-3,*,3,*,-4,*,4,$ $*,-5,*,11,*,-6,*,1,*,-11,*,9,*,+,$
4	0.43046721	$5,-2,*,2,*,-3,*,3,*,-4,*,4,*,-5,$ $*,11,*,-6,*,1,*,-13,*,10,*,-12,*,9,*,+$

The algorithm works as follows:

1) Take paths 4 and 3 from the reverse order usually choosing from longest to shortest. Enumerate those common elements, shown in series, in the center to branch out to those legs which are not common, shown in parallel to enable all paths successfully from the source (5) to the target (9) node. That is, merge

5 -2 2 -3 3 -4 4 -5 11 -6 1 -13 10 -12 9, with

5 -2 2 -3 3 -4 4 -5 11 -6 1 -11 9, to converge to:

$$\mathbf{5\ -2\ 2\ -3\ 3\ -4\ 4\ -5\ 11\ -6\ 1}\ \begin{vmatrix} -13 & 10 & -12 \\ & -11 & \end{vmatrix}\ \mathbf{9}$$

2) Then, take the next path backward 5 -1 1 -13 10 -12 9 and merge it with the RBD in 1) by following the same rule of thumb:

$$\begin{vmatrix} -13 & 10 & -12 \end{vmatrix}$$

5 -2 2 -3 3 -4 4 -5 11 -6 1 9, with

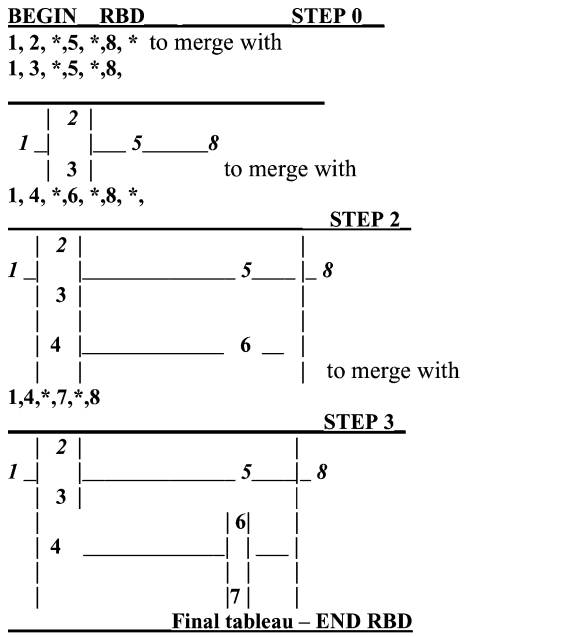
5 -1 1 -13 10 -12 9, to converge to:

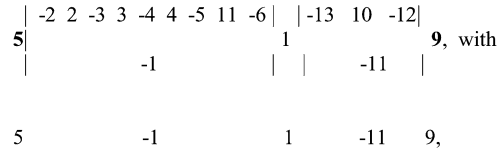
$$\mathbf{5}\begin{vmatrix} -2\ 2\ -3\ 3\ -4\ 4\ -5\ 11\ -6 \\ -1 \end{vmatrix}\ \mathbf{1}\ \begin{vmatrix} -13 & 10 & -12 \\ & -11 & \end{vmatrix}\ \mathbf{9}$$

so as to enable the passage for all paths from 5 to 9.

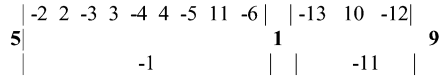
3) Finally, take the last path in the reverse order to merge with the RBD in 2)

TABLE III
ILLUSTRATION OF ALGORITHM FOR A SIMPLE SERIES–PARALLEL NETWORK

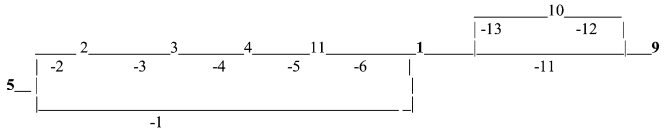




to converge to the same tableau as in 3) as the last path already existed in tableau of 2)



The final table can be expressed as in following reliability block diagram:



which results in an exact single-path reliability of 0.729, with its complete Polish notation: 5, $-1, -2$, 2, $*, -3, *$, 3, $*, -4, *$, 4, $*, -5, *$, 11, $*, -6, *, +$, 1, $*, -11, -13$, 10, $*, -12, *, +, *, *$, 9, $*$ that describes the above relationship in the final RBD.

IV. HYBRID TOOL TO COMPUTE RELIABILITY FOR COMPLEX SYSTEMS

Let us take the following non-series–parallel network to compute $s - t$ reliability, whose Boolean decomposition result is known to exist: 0.799 as in Fig. 5, Table V, [23]. With the Boolean decomposition method whose decomposed diagrams are shown (all nodes have 0.9 reliability), the system reliability is computed as follows: Node 3 bad: $\mathrm{R}(\mathrm{sys}\,|\,3\mathrm{bad}) = (0.9)[1-(1-0.81)(1-0.81)](0.9) = 0.9639(0.81) = 0.7806$. Node 3 good: $\mathrm{R}(\mathrm{sys}\,|\,3\mathrm{good}) = (0.9)[1-(1-0.9)(1-0.9)](0.9) = 0.99(0.81) = 0.8019$. Result: $\mathrm{R}(\mathrm{system}) = \mathrm{R}(3\mathrm{bad})\ \mathrm{R}(\mathrm{system}|3\mathrm{bad}) + \mathrm{R}(3\mathrm{good})\ \mathrm{R}(\mathrm{system}|3\mathrm{good}) = 0.1(.7806) + 0.9(0.8019) = 0.799$.

The system reliability is 0.799 by the "hybrid Polish-notated enumeration." After Polish-notated paths are found, remaining fictitious nodes are created to facilitate an enumeration approach. The 100+ nodes symbolize nonexistent bad nodes to denote the complement of a component, e.g., $\mathrm{R}(105) = 1 - \mathrm{R}(5)$. This hybrid method is fast for it avoids the recalculation of guaranteed paths by only calculating the probabilities of the remaining nodes' enumerated combination. This technique avoids repetition of identical combination paths. Instead of 36 paths ($2^3 = 8$ for each of the four 4-tuples and 2 for each of the two 6-tuples), we use only 18 paths saving 50%. Otherwise, the "enumeration" needs $2^7 = 128$ paths. The exact reliability using Boolean decomposition using identical nodes is $\mathrm{R}^2(4\mathrm{R}^2 - 3\mathrm{R}^3 - \mathrm{R}^4 + \mathrm{R}^5)$ and using FAST upper bound (FUB) employing the compression technique studied in [12], [19] is $\mathrm{R}^2(4\mathrm{R}^2 - \mathrm{R}^3 - 5\mathrm{R}^4 + 2\mathrm{R}^6 - \mathrm{R}^7 + 2\mathrm{R}^5)$. The theoretical difference is $\mathrm{R}^2(2\mathrm{R}^3 - 4\mathrm{R}^4 + \mathrm{R}^5 + 2\mathrm{R}^6 - \mathrm{R}^7) = 0.81 * (2 * .729 - 4 * 0.656 + 0.59 + 2 * 0.531 - 0.48) = 0.007032$. After computations shown in Fig. 6, difference (FAST upper bound − HYBRID form) $= 0.806812 - 0.799780 = 0.007032$ as expected theoretically.

We will compare FUB method's result with that of the hybrid form by using a ten-node example as in Fig. 7, where FUB reliability: 0.808879873 and HYBRID method reliability (10-Nodes) $= 0.798590485$. Difference (AST upper bound − HYBRID form) $= 0.8088798 - 0.7985905 = 0.01$. This difference is negligible for smaller networks only.

V. MORE SUPPORTING EXAMPLES FOR THE HYBRID FORM

See the eight-nodes star-like example given in Fig. 6. We will compare the approximate FUB method's result with that of the exact hybrid form. Then, a ten-node example in Fig. 7 will follow. The comparative results for the hybrid and FUB are given in Tables VI and VII.

VI. DECODING ALGORITHM

The objective is to generate a reverse coded reliability block diagram from the Polish notation and recreate the original topology generated by the RBD compression algorithm. The platform used is Java. This diagram helps view complex network paths from an ingress to an egress node, and it ultimately calculates the system reliability for series–parallel reducible networks.

The following is the approach taken to recreate the RBD from a given Polish notation.

1) Accept the Polish notation from the user. The Polish notation consists of nodes (numbers) and operators ("$*$" or "$+$").
2) Parse the Polish notation to identify the nodes and operations.
3) Identify the node pairs that connect. Use the existing Java components and the node pairs that are identified to draw the RBD.

TABLE IV
EXACT RELIABILITIES FOR THE SERIES–PARALLEL DING-DONG1 AS IN FIG. 3 (SCREENS 3 AND 4)

Analytical Results: Ingress Node: 1 Egress Node: 13
Without Transmission (Link) Reliabilities, Exact Reliability: 0.7938938796628517
Polish Notation: 1, 2, 4, *,5,* 6,* 7,*,8, 9,10,*, +, *, 11,*, 3,14, *,15, *,16, *,17,18, +, *,19, *, +,12, +, *,13, *

Path #	Reliability	Polish Notation
1	0.38742048	1, 2, *,4, *,5, *,6, *,7, *,8, *,11, *,13, *,
2	0.34867844	1, 2, *,4, *,5, *,6, *,7, *,9, *,10, *,11, *,13, *, +,
3	0.43046721	1, 3, *,14, *,15, *,16, *,17, *,19, *,13, *, +,
4	0.43046721	1, 3, *,14, *,15, *,16, *,18, *,19, *,13, *, +,
5	0.72900000	1, 12, *,13, *

With Transmission (Link) Reliabilities, Exact Reliability: 0.5513297153873634
Polish Notation: 1, -1, 2, *, -2, *, 4, *, -3, *, 5, *, -4, *, 6, *, -5, *, 7, *, -6, 8, *, -10, *, -7, 9, *, -8, *, 10, *, -9, *, +, *, 11, *, -21, *, -11, 3, *, -12, *, 14, *, -13, *, 15, *, -14, *, 16, *, -15, 17, *, -17, *, -16, 18, *, -18, *, +, *, 19, *, -22, *, +, -19, 12, *, -20, *, +, *, 13, *

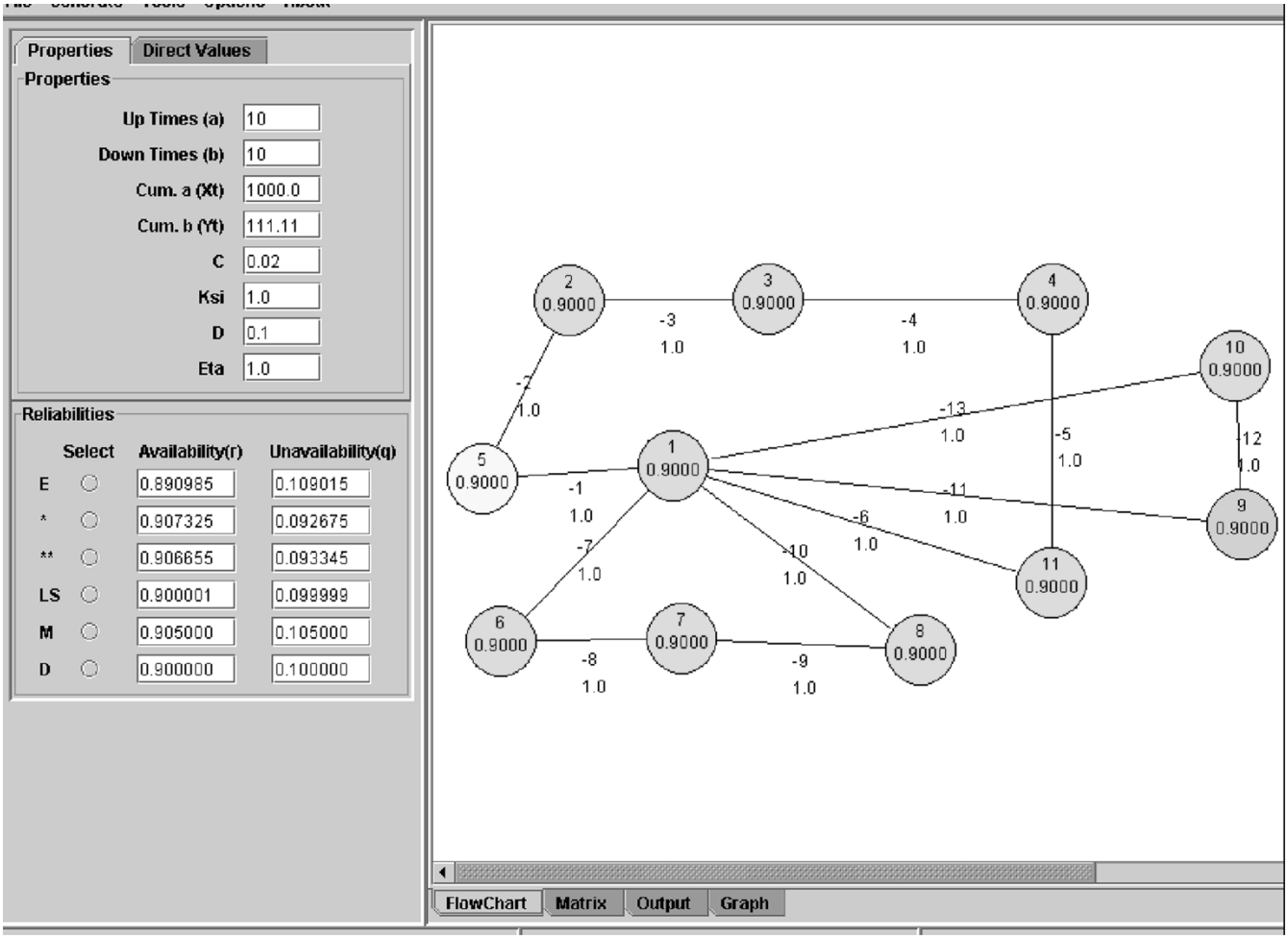


Fig. 4. Screen 5: 11-node network. Source (ingress) is node 5 and the target (egress) is node 9.

A stack algorithm was employed to accomplish the above. The algorithm accepts the Polish notation and parses the notation using Java's string tokenizer. To identify the node pairs that connect, the following logic was incorporated:

- Push into the stack until an operator is encountered.
- If the operator is a "$*$" (nodes in series),
  Pop the top two elements (nodes) of the stack.
  Form a node pair.
  Concatenate the nodes and node pairs.
  Push the concatenated string onto the top of the stack.
- If the operator is a "+" (nodes in parallel)
  Pop the top two elements (nodes) of the stack.
  Concatenate the operator between the two nodes.
  Push the concatenated string on to the top of the stack.
- Continue performing the above steps until the end of the Polish notation.

After the node pairs are identified, the graphical Java components FC oval (nodes) and FC Line (transmissions or connecting links) display the network.

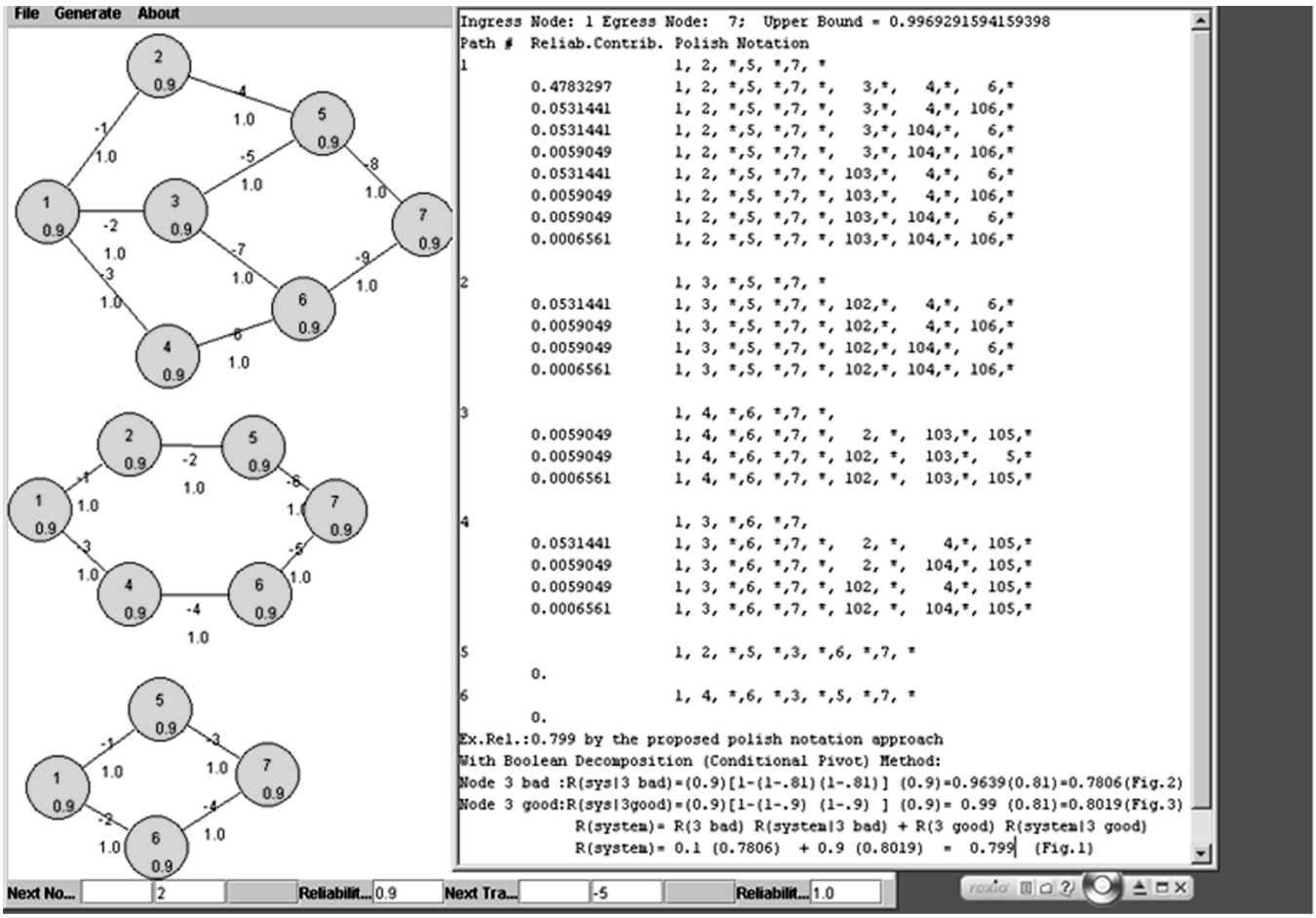


Fig. 5. Screen 6: RBD for 7-Node Network. Table V: Exact $s-t$ reliability $(\mathrm{s}=1, \mathrm{t}=7)$ for a non-series-parallel network with a Boolean decomposition.

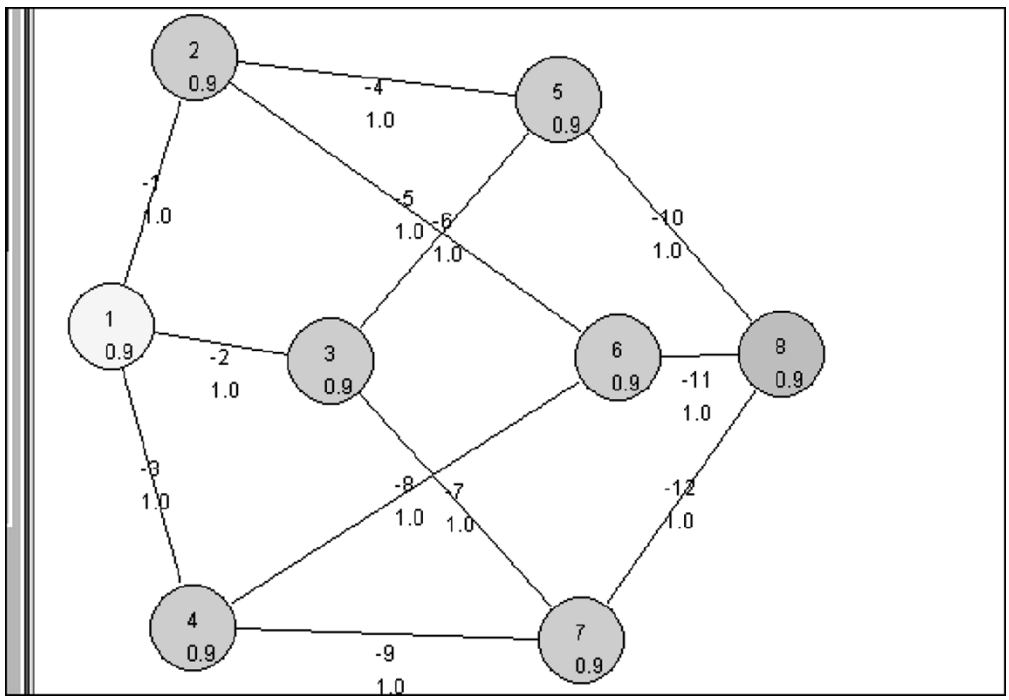


Fig. 6. Screen 7: RBD for an eight-node network for $\mathrm{s}=1,\ \mathrm{t}=8$.

Networks utilizing links were deployed using the same algorithmic process. Negative digits, which designate transmission lines, will be represented as nodes first. Once the initial diagram has been generated, a second process will, essentially, remove

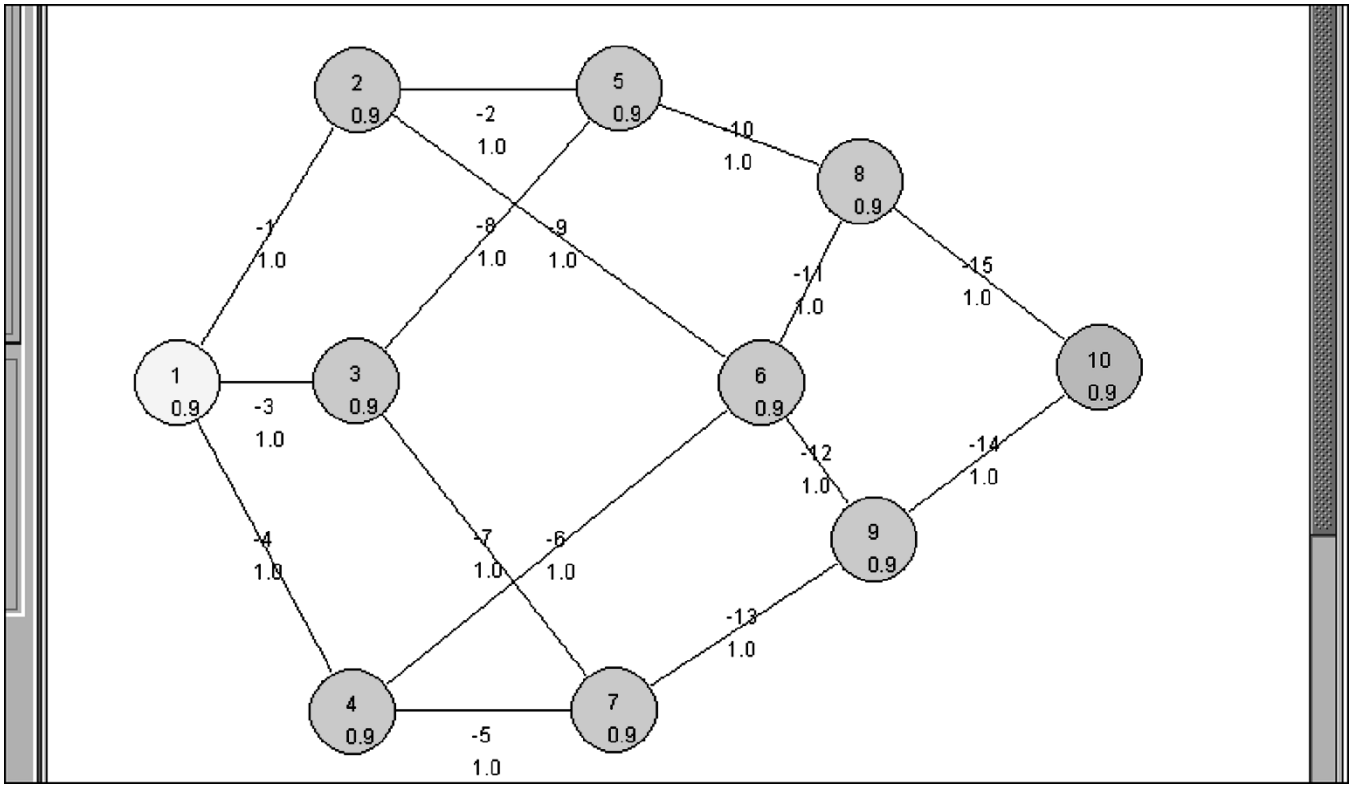


Fig. 7. Screen 8: RBD for a ten-node network for $s = 1,\ t = 10$.

TABLE VI
EXACT $s - t$ RELIABILITY FOR THE COMPLEX EIGHT-NODE IN FIG. 7.

Ingress Node:1, Egress Node: 8; Network Reliability(FAST Upper Bound)Method= 0.80895 (No Link Failures). Polish Notation: *1, -1, 2, *, -4, 5, *, -6, 3, *, -7, *, 7, *, -9, 4, *, -8, *, 6, *, -11, *, -12, +, *, -10, +, *, -5, 6, *, -8, 4, *, -9, *, 7, *, -7, 3, *, -6, *, 5, *, -10, *, -12, +, *, -11, +, *, +, *, -2, 3, *, -6, 5, *, -4, 2, *, -5, *, 6, *, -8, 4, *, -9, *, 7, *, -12, *, -11, +, *, -10, +, *, -7, 7, *, -9, 4, *, -8, *, 6, *, -5, 2, *, -4, *, 5, *, -10, *, -11, +, *, -12, +, *, +, *, +, -3, 4, *, -8, 6, *, -5, 2, *, -4, *, 5, *, -6, 3, *, -7, *, 7, *, -12, *, -10, +, *, -11, +, *, -9, 7, *, -7, 3, *, -6, *, 5, *, -4, 2, *, -5, *, 6, *, -11, *, -10, +, *, -12, +, *, +, *, +, *, 8 *
Exact HYBRID (8-Nodes) = 0.80818398;
Difference (FAST Upper Bound-HYBRID Form) = 0.80895 - 0.80818 = 0.00077

the oval object, which represents a node, leaving the negative node name as the transmission line. The smallest node number is the ingress; the largest one is the egress node.

A more complex non-series-parallel telephony network (with 19 nodes and 32 links) whose Polish-notation was previously coded is reverse-coded or decoded in Fig. 8 to reconstruct the original topology. Note that the hard-to-read "Polish Notation" box is a page-long Polish notation previously obtained and inserted using the "compression" algorithm. Although the said Polish notation does not calculate the exact $s - t$ reliability for non-series-parallel networks (for which a specific hybrid technique has been demonstrated in the preceding sections), it does successfully encode and decode any non-series–parallel network for a secure and economical transport. The Polish notation approach also prepares a base for calculating the exact reliability for any complex system utilizing a hybrid enumeration approach. The following Fig. 8 denotes all nodes and links invariably with a given sample reliability of 0.99 for simplicity and convenience. The node and link reliabilities can be altered at will. The ingress and egress are $s = 1$ and $t = 19$, respectively.

VII. CONCLUSION AND FUTURE STUDIES

Primarily, the proposed hybrid algorithm calculates the exact $s - t$ reliability index starting from simpler and more tractable complex networks such as in Figs. 6 and 7, and further to more complex as in Fig. 8. Second, the compression technique used in FUB method, which does not give exact results but an approximate fast upper bound, however, performs a special coding to encode and decode non-series–parallel (complex) networks. Fig. 8 with 19 nodes and 32 links illustrates how to reconstruct the given complex topology by a special conversion or decoding technique. The algorithm is given in Section VI. This practice can be useful for security, and time- and space-saving purposes. This package enables encoding and decoding for any network of higher and critical assurance.

In conclusion, aside from calculating the $s - t$ reliability of any complex system, it is shown that the Polish-notation constructed from the graphical interface using post-fixes to describe the topology of any complex network is useful for identifying a given topology. Furthermore, the output can then be transported, for reasons of security or saving storage space, to a

TABLE VII
EXACT $s-t$ RELIABILITY FOR THE COMPLEX TEN-NODE IN FIG. 8.

Ingress Node: 1, Egress Node: 10; Network Reliability (FAST Upper Bound) Method= 0.80887 (No Link Failures). Polish Notation: 1, -1, 2, *, -2, 5, *, -8, 3, *, -7, *, 7, *, -5, 4, *, -6, *, 6, *, -11, 8, *, -15, *, -12, 9, *, -14, *, +, *, -13, 9, *, -12, 6, *, -11, *, 8, *, -15, *, -14, +, *, +, *, -10, 8, *, -11, 6, *, -6, 4, *, -5, *, 7, *, -13, *, -12, +, *, 9, *, -14, *, -15, +, *, +, *, -9, 6, *, -6, 4, *, -5, *, 7, *, -7, 3, *, -8, *, 5, *, -10, *, 8, *, -15, *, -13, 9, *, -14, *, +, *, -11, 8, *, -10, 5, *, -8, *, 3, *, -7, *, 7, *, -13, *, 9, *, -14, *, -15, +, *, +, -12, 9, *, -13, 7, *, -7, *, 3, *, -8, *, 5, *, -10, *, 8, *, -15, *, -14, +, *, +, *, +, *, -3, 3, *, -7, 7, *, -5, 4, *, -6, *, 6, *, -9, 2, *, -2, *, 5, *, -10, *, -11, +, 8, *, -15, *, -12, 9, *, -14, *, +, *, -13, 9, *, -12, 6, *, -9, 2, *, -2, *, 5, *, -10, *, -11, +, *, 8, *, -15, *, -14, +, *, +, *, -8, 5, *, -2, 2, *, -9, *, 6, *, -6, 4, *, -5, *, 7, *, -13, *, -12, +, 9, *, -14, *, -11, 8, *, -15, *, +, *, -10, 8, *, -11, 6, *, -6, 4, *, -5, *, 7, *, -13, *, -12, +, *, 9, *, -14, *, -15, +, *, +, *, +, *, +, -4, 4, *, -5, 7, *, -7, 3, *, -8, *, 5, *, -2, 2, *, -9, *, 6, *, -11, 8, *, -15, *, -12, 9, *, -14, *, +, *, -10, 8, *, -11, 6, *, -12, *, 9, *, -14, *, -15, +, *, +, *, -13, 9, *, -12, 6, *, -9, 2, *, -2, *, 5, *, -10, *, -11, +, *, 8, *, -15, *, -14, +, *, +, *, -6, 6, *, -9, 2, *, -2, *, 5, *, -8, 3, *, -7, *, 7, *, -13, *, 9, *, -14, *, -10, 8, *, -15, *, +, *, -11, 8, *, -10, 5, *, -8, *, 3, *, -7, *, 7, *, -13, *, 9, *, -14, *, -15, +, *, +, -12, 9, *, -13, 7, *, -7, *, 3, *, -8, *, 5, *, -10, *, 8, *, -15, *, -14, +, *, +, *, +, *, +, *, 10, * Exact HYBRID (10-Nodes) = 0.79859 Difference (FAST Upper Bound-HYBRID Form) = 0.80887- 0.79859= 0.001

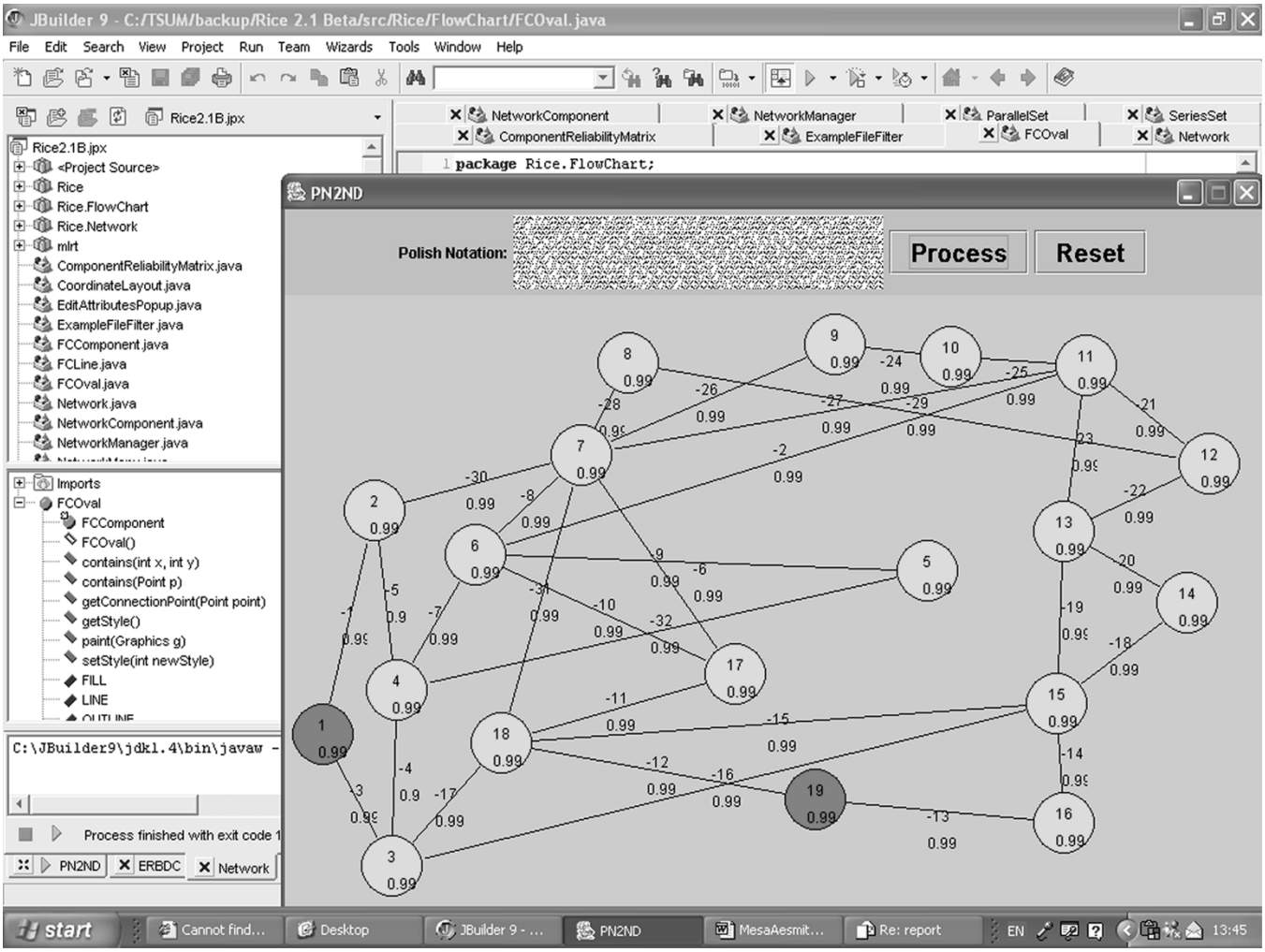


Fig. 8. Screen 9: Decoding (reverse Polish-notation) Process for the complex 32-node network.

remote analyst, who in turn using this algorithm can reverse a given Polish-notation in a decoding phase to reconstruct the topology. Both forward (encoding) and reverse (decoding) algorithms work for both simple series–parallel and non-series–parallel, i.e., complex networks. Further, a proposed hybrid method also computes the exact $s-t$ reliability for complex systems as opposed to an approximate fast upper bound (FUB) more suitable for simple series–parallel networks because it is considerably faster and easier. However, FUB compromises the accuracy as compared to the hybrid method for complex networks. Networks of various complexities are examined. The efforts to save time are in progress for large networks exceeding 20 nodes.

The proposed hybrid algorithm can accurately calculate the $s-t$ reliability of a complex system such as in Fig. 8 with 19 nodes in roughly 3000 s. A new research project using an overlap technique is in its final stages to increase the computa-

tion speed for large complex networks in the order of a thousand fold, from 3000 to 3 s, without sacrificing any accuracy for the Fig. 8 network of 19 nodes and 32 links. The new approach with an extreme speed and additional advantages such as multistate treatment of the components is the subject matter of a new publication in the near future.

ACKNOWLEDGMENT

The authors would like to thank J. Larson, B. Rice, S. Reddy, and J. Moon from Troy University in Montgomery for their programming assistance.

REFERENCES

[1] C. J. Colbourn, "Combinatorial aspects of network reliability," *Ann. Operations Res.*, vol. R-30, pp. 32–35, 1981.

[2] K. K. Aggarwal and S. Rai, "Reliability evaluation in computer communication networks," *IEEE Trans. Rel.*, vol. R-30, no. 1, pp. 32–35, Mar. 1981.

[3] R. H. Jan, "Design of reliable networks," *Comput. Operations Res.*, vol. 20, no. l, pp. 25–34, 1993.

[4] M. S. Yeh, J. S. Lin, and W. C. Yeh, "New Monte Carlo method for estimating network reliability," in *Proc. 16th Int. Conf. Computers and Industrial Engineering*, 1994, pp. 723–726.

[5] G. S. Fishman, "A comparison of four Monte Carlo methods for estimating the probability of $s - t$ connectedness," *IEEE Trans. Rel.*, vol. R35, no. 2, pp. 145–155, Jun. 1986.

[6] M. Sahinoglu and D. Libby, "Sahinoglu–Libby (SL) probability density function-component reliability applications in integrated networks," in *Proc. Ann. Reliability and Maintainability Symp.*, Tampa, FL, Jan. 27–30, 2004, pp. 280–287. RAMS03.

[7] B. Dengiz, F. Altiparmak, and A. E. Smith, "Efficient optimization of all-terminal reliable networks using evolutionary approach," *IEEE Trans. Rel.*, vol. 46, no. 1, pp. 18–26, 1997.

[8] B. Dengiz, F. Altiparmak, and A. E. Smith, "Local search genetic algorithm for optimal design of reliable networks," *IEEE Trans. Evol. Comput.*, vol. 1, no. 3, pp. 179–188, Sep. 1997.

[9] M. Sahinoglu, *Reliability Index Evaluations of an Integrated Software System for Insufficient Software Failure and Recovery Data*. Izmir, Turkey: Springer-Verlag, Oct. 2000, pp. 25–27.

[10] K. E. Murphy and C. M. Carter, "Reliability block diagram construction techniques: Secrets to real-life diagramming woes," in *Proc. Ann. Reliability and Maintainability Symp.—Tutorial Notes RAMS03*, Tampa, FL, Jan. 2003.

[11] L. C. Woltenshome, *Reliability Modeling—A Statistical Approach*. London, U.K.: Chapman & Hall/CRC, 1999, pp. 106–107.

[12] M. Sahinoglu, J. Larson, and B. Rice, "An exact reliability calculation tool to improve large safety-critical computer networks," in *Proc. DSN2003*. San Francisco, CA, Jun. 22–25, 2003, pp. B-38–B-39.

[13] T. Luo and K. S. Trivedi, "An improved algorithm for coherent system reliability," *IEEE Trans. Rel.*, vol. 47, no. 1, pp. 73–78, Mar. 1998.

[14] S. Rai, M. Veeraraghavan, and K. S. Trivedi, "A survey on efficient computation of reliability using disjoint products approach," *Networks*, vol. 25, no. 3, pp. 174–163, 1995.

[15] X. Zang, H. R. Sun, and K. S. Trivedi, "A BDD approach to dependable analysis of distributed computer systems with imperfect coverage," in *Dependable Network Computing*, D. Avresky, Ed. Amsterdam, The Netherlands: Kluwer, 1999, pp. 167–190.

[16] H. Sun, X. Zang, and K. S. Trivedi, "A BDD based algorithm for reliability analysis of phase mission systems," *IEEE Trans. Rel.*, vol. 48, no. 1, pp. 50–60, Mar. 1999.

[17] M. Sahinoglu, "An exact RBD calculation tool to design very complex systems," in *Proc. 1st ACIS Int. Conf. Software Engineering Research Applications*, San Francisco, CA, Jun. 25–27, 2003. Invited talk.

[18] C. V. Ramamoorthy and Y. W. Han, "Reliability analysis of systems with concurrent error detection," *IEEE Trans. Comput.*, vol. 26, pp. 868–878, Sep. 1975.

[19] M. Sahinoglu, A. Smith, and B. Dengiz, "Improved network design method when considering reliability and cost using an exact reliability block diagram calculation (ERBDC) tool in complex systems," in *Proc. Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 13, St. Louis, MO, Nov. 1–4, 2003, pp. 849–855. ANNIE-Smart Engineering Systems.

[20] M. Sahinoglu, C. V. Ramamoorthy, A. Smith, and B. Dengiz, "A reliability block diagramming tool to describe networks," in *Proc. Ann. Reliability and Maintainability Symp.*, Los Angeles, CA, Jan. 26–29, 2004, pp. 141–145. RAMS04.

[21] M. Sahinoglu and W. Munns, "Availability indices of a software network," in *Proc. 9th Brazilian Symp. Fault Tolerant Computing*, Florianopolis, Brazil, Mar. 2001, pp. 123–131.

[22] M. Sahinoglu, "An algorithm to code and decode complex systems, and to compute $s - t$ reliability," in *Proc. Ann. Reliability and Maintainability Symp.*, Alexandria, VA, Jan. 24–27, 2005, pp. 67–70. RAMS05.

[23] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, 2nd ed, New York: Wiley, 2002, pp. 42–60.

[24] M. Sahinoglu, "An empirical Bayesian stopping rule in testing and verification of behavioral models," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 5, pp. 1428–1443, Oct. 2003.

[25] S. R. Das, M. Sudarma, M. Assaf, E. Petriu, W. B. Jone, and M. Sahinoglu, "Parity bit signature in response data compaction and built-in self-testing of VLSI circuits with nonexhaustive test sets," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 5, pp. 1363–1380, Oct. 2003.

[26] S. R. Das, M. Assaf, E. Petriu, and M. Sahinoglu, "Aliasing-free compaction in testing cores-based system-on-chip (SOC), using compatibility of response data outputs," *Trans. SDPS*, vol. 53, no. l, pp. 1–17, Mar. 2004.

[27] M. Sahinoglu, D. Libby, and S. R Das, "Measuring availability indices with small samples for component and network reliability using the Sahinoglu–Libby probability model," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 3, pp. 1283–1295, Jun. 2005.

[28] S. R. Das, C. V. Ramamoorthy, M. Assaf, E. Petriu, W. B. Jone, and M. Sahinoglu, "Fault simulation and response compaction in full-scan circuits using HOPE," in *Proc. 19th IEEE Instrumentation and Measurement Technology Conf.*, vol. 1, May 2002, pp. 607–612.

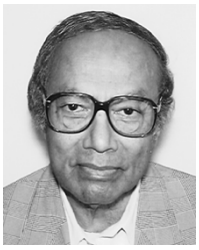
[29] ——, "Revisiting response compaction in space for full scan circuits with nonexhaustive test sets using concept of sequence characterization," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 5, pp. 1662–1677, Oct. 2005.

[30] M. Sahinoglu, "Security meter—A practical decision tree model to quantify risk," *IEEE Security Privacy*, vol. 3, no. 3, pp. 18–24, May/Jun. 2005.

Mehmet Sahinoglu (S'78–M'81–SM'93) received the B.S. degree from the Middle East Technical University (METU), Ankara, Turkey, and the M.S. degree from the University of Manchester Institute of Science and Technology (UMIST), Manchester, U.K., both in electrical and computer engineering, and the Ph.D. degree in electrical engineering and statistics from Texas A&M, College Station.

He is the Eminent Scholar for the Endowed Chair of the Alabama Commission of Higher Education (ACHE) and Chairman of the Computer and Information Science (CIS) at Troy State University, Montgomery, AL, since 1999. Following his 20-year-long tenure at METU as an Assistant/Associate/Full Professor, he served as the First Dean, and the Founding Department Chair in the College of Arts and Sciences at DEU, Izmir, Turkey, from 1992 to 1997. He was a Chief Reliability Consultant to the Turkish Electricity Authority (TEK) from 1982 to 1997. He became an Emeritus Professor at METU and DEU in 2000. He lectured at Purdue University, West Lafayette, IN (1989–1990, 1997–1998) and Case Western Reserve University, Cleveland, OH (1998–1999) in the capacity of a Fullbright, and a NATO Scholar, respectively.

Dr. Sahinoglu is a Fellow of the Society of Design and Process Science (SDPS), a Member of ACM, AFCEA, and ASA and an Elected Member of ISI. He is accredited for the "Compound Poisson Software Reliability Model" to account for the multiple (clumped) failures in predicting the total number of failures at the end of a mission time, and the "MESAT: Compound Poisson Stopping Rule Algorithm" in cost-effective digital software testing. He is also jointly responsible with Dr. D. Libby for the original derivation of the Generalized Three-Parameter Beta (G3B) pdf in 1981, also known as Sahinoglu and Libby (SL) pdf in 1999.



Chittoor V. Ramamoorthy (M'57–SM'76–F'78 –LF'93) received degrees in physics and technology from the University of Madras, Madras, India, two graduate degrees in mechanical engineering from the University of California, Berkeley, and the A.M. and Ph.D. degrees in electrical engineering and computer science (applied mathematics) from Harvard University, Cambridge, MA, in1964. His education at Harvard was supported by Honeywell Inc. with whom he was associated last as a Senior Staff Scientist.

He later joined the University of Texas, Austin, as a Professor in the Department of Electrical Engineering and Computer Science. After serving as Chairman of the Department, he joined the University of California, Berkeley, in 1972 as Professor of Electrical Engineering and Computer Sciences, Computer Science Division, a position that he still holds as Professor Emeritus. He has supervised more than 70 doctoral students in his career. He has held the Control Data Distinguished Professorship at the University of Minnesota, Minneapolis, and Grace Hopper Chair at the U.S. Naval Postgraduate School, Monterey, CA. He was also a Visiting Professor at Northwestern University, Evanston, IL, and a Visiting Research Professor at the University of Illinois, Urbana-Champaign. He is a Senior Research Fellow at the ICC Institute of the University of Texas. He has published more than 200 papers and has also coedited three books. He has worked on and holds patent in computer architecture, software engineering, computer testing and diagnosis, and in databases. He is currently involved in research on models and methods to assess the evolutionary trends in information technology. He is also the founding Co-Editor-in-Chief of the *International Journal of Systems Integration* and the *Journal for Design and Process Science*.

Dr. Ramamoorthy received the Group Award and Taylor Booth Award for education from the IEEE Computer Society, the Richard Merwin Award for outstanding professional contributions, and the Golden Core Recognition, and is the recipient of the IEEE Centennial Medal and the IEEE Millennium Medal. He also received the Computer Society's Kanai–Hitachi Award for the year 2000 for pioneering and fundamental contributions in parallel and distributed computing and the Best Paper Award in 1987. He was the cowinner (with Prof S. Das of Troy State University) of the IEEE's Best Paper Award—the Donald Fink Prize Paper Award for 2003 for their paper published in the December 2001 issue of the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT. He is a Fellow of the Society of Design and Process Science from which he received the R. T. Yeh Distinguished Achievement Award in 1997. He also received the Best Paper Award from the IEEE Computer Society in 1987. Three international conferences were organized in his honor, and one UC Berkeley Graduate Student Research Award, and two International Conferences/Society Awards have been established in his name. and in 2002, he received its Gold Medal of Honor. He served as the Editor-in-Chief of the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. He is the founding Editor-in-Chief of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, which recently published a Special Issue in his honor. He served in various capacities in the IEEE Computer Society including its First Vice President, and a Governing Board Member. He served on several Advisory Boards of the Federal Government and of the Academia that include the United States Army, Navy, Air Force, DOEs, Los Alamos Lab, University of Texas, and State University System of Florida. He is one of the founding Directors of the International Institute of Systems Integration in Campinas, Brazil, supported by the Federal Government of Brazil, and for several years, was a Member of the International Board of Advisors of the Institute of Systems Science of the National University of Singapore, Singapore.