FAULT DIAGNOSIS OF THE APS REAL-TIME ORBIT FEEDBACK SYSTEM BASED ON FTA *

S. Xu#, H. Shang, F.R. Lenkszus, R. Laird, C. Yao, Argonne National Laboratory, Argonne, IL 60439, U.S.A.

Abstract
The Advanced Photon Source (APS) real-time orbit feedback system is complex, and faults are difficult to diagnose. This paper presents a diagnostic method based on fault tree analysis (FTA). The fault tree is created based on more than ten years of operating experience on the system. The method is described to analyze the fault tree. The operator interface to the diagnostic tool is discussed.

INTRODUCTION
The Advanced Photon Source (APS) is the foremost third-generation synchrotron light source in the western hemisphere, delivering intense x-ray to as many as 35 insertion devices and 35 bending magnet beamlines. Orbit stability is critical to achieving optimum performance for the x-ray users. The APS real-time orbit feedback system plays an important role in maintaining the orbit stability and is very complex. The orbit feedback hardware is distributed around the 1.1-km storage ring with a total of 20 slave VME crates accessing beam position monitors (BPMs) and correctors and performing orbit feedback calculations. A separate master VME crate controls the slaves and performs real-time data analysis [1]. Fast real-time data transfer between these crates is implemented through a dedicated reflective memory network. All the crates are also connected to the APS controls network for slow remote control and data monitoring.

The user interface for the real-time orbit feedback system provides some indicators on the system status, but there are no existing tools to produce a diagnostic procedure when the system fails. This paper presents a method for producing a diagnostic procedure based on fault tree analysis.

PRINCIPLE OF THE IMPLEMENTATION
Fault tree analysis is a top-down deductive analytical method to evaluate system reliability and safety. It can be qualitative or quantitative, depending on whether failure event probabilities are unknown or known. On the fault tree, fault basic components are connected through logic gates to the top event. Based on the system's fault tree and the basic components' failure probability knowledge, the system's diagnosis decision tree can be built: the Binary Decision Diagram (BDD) can be created from the fault tree; a Diagnostic Importance Factor (DIF) for each basic component can be calculated from the BDD; and the diagnostic model can be constructed by DIF ordering [2].

A BDD was introduced to represent a Boolean function as a decision graph. It is based on Shannon decomposition, which is defined in the form of an If-Then-Else (ITE) connective

\[
f = \text{ite}(e, f_1, f_0) = ef_1 + \bar{e}f_0,
\]

where \( f \) is a Boolean function, and \( e \) is the variable. The function \( f_1 \) is evaluated with \( e=1 \); \( f_0 \) is evaluated with \( e=0 \). By recursively applying the Shannon decomposition over all the variables, the Boolean function \( f \) can be graphically represented as a binary tree.

Once the BDD is constructed, two importance factors for each basic element the Marginal Importance Factor (MIF) and the DIF can be calculated according to the following equations [2,3]:

\[
\begin{align*}
MIF(S,e) &= p(S/e) - p(S/\bar{e}), \\
DIF(S,e) &= p(e/S), \\
DIF(S,e) &= \frac{p(e)p(\bar{e})MIF(S,e)}{p(S)}.
\end{align*}
\]

The MIF\((S,e)\) is the difference in unreliability of the system \( S \) with component \( e \) failed, \( p(S/e) \) and the unreliability of the system \( S \) with component \( e \) functioning, \( p(S/\bar{e}) \). The DIF\((S,e)\) measures the probability that a certain component \( e \) has failed when the system \( S \) fails [2]. The diagnosis procedure can be produced by calculating the DIFs for all basic components.

IMPLEMENTATION
Fault Tree Creation
There are a lot of commercially available tools for creating a fault tree. An open source tool, openFTA [4], was used to generate the fault tree for the APS real-time orbit feedback system. An internal database file for each fault tree is used to store basic event information. Each basic event has the following attributes: id, probability, type, and description. Middle event, basic event, and its probability were generated by more than ten years operations experience at the APS. The APS Controls group maintains a logbook in which many kinds of fault events and their occurrences, observations, actions, and guidance are recorded. The overall fault tree for the real-time orbit feedback system is shown in Figure 1.

Based on the fault tree information, a fault tree database in SDDS [5] format was generated by a Tcl script. Each


#xusf@aps.anl.gov
The page of the database represents the elements of one subtree. The database’s parameters describe the properties of the subtree. The parameters are:

- **Label** -- the unique label of the subtree.
- **ID** -- the ID of the subtree; each ID is unique and has a corresponding label.
- **LogicalType** -- the logical type of the subtree, either “OR” or “AND”. It represents the relationship of the elements of the subtree.
- **Description** -- the description of the subtree.

The elements of the subtree are represented by rows and columns. The number of rows represents the number of elements in the subtree, and the columns represent the properties of each element. The columns are:

- **ID** -- a unique number that represents the elements. If it is greater than 1000, the element is a base element; otherwise, it is also a subtree and its properties will be shown in subsequent pages.
- **Label** – a unique label for the element.
- **Description** -- a description of the element.
- **Probability** -- the fault probability of the element.
- **Guidance** -- guidance for checking and fixing the fault elements.

The database can be edited by a Tcl/Tk tool sddsedit, as shown in Figure 2.

**Algorithm Implementation**

The core algorithm was implemented by a C routine convert_to_bdd(). It reads the fault tree SDDS database, converts the fault tree into BDD, and then calculates the DIF for each basic element of all failed subtrees. It can also displays the BDD map of subtrees.

The conversion from fault tree to BDD involves the recursive operation of two ITE connectives: AND and OR. The calculation of DIF involves computing MIF and system failure probability \( p(S) \). These are also implemented by recursive subroutines.

The algorithm can also incorporate evidence into the DIF calculation. This means it can accept a list of parameters that shows some basic elements are known failed \( p(e)=1 \) and some basic elements are known no-failure \( p(e)=0 \).

**User Interface**

The user interface for the diagnosis of the APS storage ring real-time feedback system is shown in Figure 3. It was created in Tcl/Tk, and can display the fault tree structure including subtrees and basic elements. It can operate in two modes: online or offline. For online operation, some subtrees and basic elements have process variables (PVs) linked to the real system indicating their failure status. For offline operation, a user has to select which parts have failed, either subtrees or top tree. Once the diagnosis command is issued, the diagnosis
Figure 3: User interface for real-time feedback system diagnostic tool.

Figure 4: Guidance window.

decision tree will be generated. A series of windows will pop up, one by one, showing the guidance for checking or repairing each possible failure basic element. Figure 4 shows one of the guidance windows.

SUMMARY

This paper presents a direct way to diagnose faults in the APS real-time orbit feedback system. This should help reduce the MTTR of the whole APS. The offline mode of the user interface has been tested. The online mode will be tested as time becomes available. The failure probability for each basic element of the fault tree is completely determined by past experience; this probability may need to be adjusted in the future as further experience is gained.

REFERENCES