# On Fault Diagnosis using Bayesian Networks ; A Case Study of Combinational Adders.

Sara Zermani, Catherine Dezan, Reinhardt Euler

**Research report, February 2014**
(Master project from February 2013 to August 2013)

# On Fault Diagnosis using Bayesian Networks ; A Case Study of Combinational Adders.

Sara Zermani, Catherine Dezan, Reinhardt Euler

Lab-STICC, CNRS UMR 6285,
Université de Bretagne Occidentale,
20 Avenue Victor Le Gorgeu
29238 Brest - France
Sara.Zermani@univ-brest.fr

**Abstract**
In this paper, we use Bayesian networks to reduce the set of vectors for the test and the diagnosis of combinational circuits. We are able to integrate any fault model (such as bit-flip and stuck-at models) and consider either single or multiple faults.
We apply our method to adders and obtain a minimum set of vectors for a complete diagnosis in the case of the bit-flip model. A very good diagnosis coverage for the stuck-at fault model is found with a minimum set of test vectors and a complete diagnosis by adding few vectors.

**Key words:** Bayesian inference, fault models, diagnosis, optimizations, adders.

## 1. Introduction

In recent years, the development of integrated circuit technology has accelerated rapidly. Accordingly, digital systems are built with more and more complexity. Fault diagnosis of circuits becomes an important and necessary part of the validation process, especially in the context of nanotechnology, where the fault rate becomes important.
The classical diagnosis algorithms are based on either a "Cause-to-Effect" approach or an "Effect-to-Cause" approach [1]. The first one is based on a process of fault simulation. It generates a fault dictionary containing the responses of the circuit to a test sequence in the presence of a set of given faults [2]. The diagnosis is executed by comparing the responses of the circuit under test with those stored in the dictionary. The second approach uses a search process of failure position from information produced on the outputs of the circuit during the test application. This approach can be based on a process of critical path tracing (CPT) [1].
Recently, other methods of diagnosis combining both approaches have been presented in the literature [3]. In addition, some diagnostic processes are based on the generation of a new type of test vectors called distinction vectors. In this case, the result provided by the logic diagnosis is used as input to a process of automatic generation of test vectors (ATPG, Automatic Test Pattern Generation) allowing to generate additional test vectors that are able to discriminate potential failures and thus to improve the diagnostic resolution [4] [5].
Bayesian network diagnoses are commonly based on models which exploit probabilities. The use of Bayesian inference can be viewed as a combination of both the "Effect-to-Cause" and the "Cause-to-Effect" approaches integrating a probability model. Many applications of Bayesian networks deal with the diagnosis of a limited number of representative elements of the system.
For example, at the system level, the combination of genetic optimization with Bayesian networks provides a robust and powerful system-design tool [6]. At the architecture level, an approach of error modeling for nano-domain logic circuits using Bayesian networks is found in [7]. Bayesian networks have also been used for the diagnosis of the layout level [8].

The objective and originality of this paper is to demonstrate the interest of Bayesian networks for the field of diagnosis of combinational circuits by integrating fault models to minimize the set of diagnosis vectors. Specific strategies are applied to regular arithmetic circuits like adders.
The design of arithmetic structures with error detection and correction capabilities represents an important research topic, and since adders are essential building blocks in all arithmetic processing units, we have used them as our study case.
In the first section of this paper, we present the Bayesian model for fault test and diagnosis and we show how the malfunctioning features can be improved by introducing logic fault models at different hierarchical levels. In the second section, we present an original algorithm for test and diagnosis with potential optimizations. We also present the adaptation of the algorithm to the hierarchical model. In the last section, we conclude with the presentation of different simulation results that we obtained with BNT (Matlab Toolbox for Bayesian Networks) applied to adders. As a result, we obtain a minimum set of diagnosis vectors with a very good diagnosis coverage.

## 2. A Bayesian fault model for test and diagnosis

### 2.1. Bayesian networks and combinational circuits

Bayesian networks are at the intersection of graph theory and probability theory. The system is modeled by a directed acyclic graph (DAG), in which the nodes represent random variables and arcs represent conditional dependencies between variables. The probability space is a set of a-priori (no direct cause) or conditional probabilities (one or more direct causes) associated with network variables. The Bayesian inference is to propagate certain information (evidences made on the system) in the network and to calculate the conditional probability of events connected to each other based on Bayes' rule [9].
From the structural description of a combinational circuit, we can build a Bayesian network which expresses its functionality. Each input of the circuit is represented by a node having an a-priori probability table (at random or according to the context of use). Each component of the structure is also represented by a node together with a conditional probability table expressing its operation.
Figure 1 illustrates an example of a Bayesian network representing an adder circuit. Using Bayesian inference, we can calculate the probabilities for all nodes by giving a few observations. If we have observed the value 0 at both the circuit outputs OR and XOR2, we can deduce the probabilities of the inputs Ret0, A1 and B1 using Bayesian inference. We obtain P(A1=0)=1, P(B1=0)=1 and P(Ret0=0)=1 which means A1=0, B1=0 and Ret0=0.

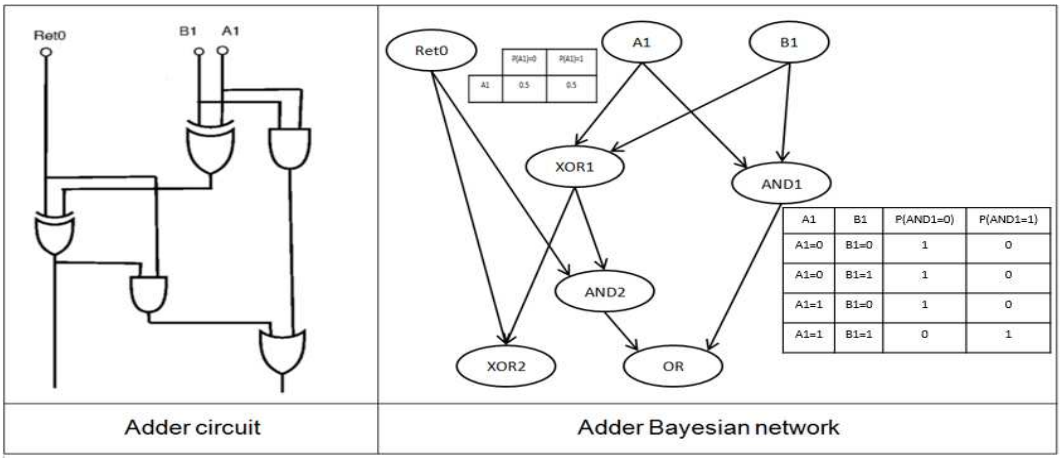



FIGURE 1 – An example of a Bayesian network

### 2.2. The Bayesian network of a fault model

To simulate and study the malfunctioning of a circuit, we introduce several fault models in the Bayesian representation. In our case, the bit-flip and the stuck-at fault model are experimented, but other types could be introduced such as "open circuit", "bridge", etc.
According to the fault model, the values are evolving as follows :
- in the Bit-flip model, the value of the output is inverted permanently ;
- in the Stuck-at model, the value of the output/input is permanently fixed to 0 or 1.
Both models can be applied to single or multiple faults.
They are integrated into the Bayesian network of the circuit as follows :

1. At the "graphical level", new nodes are added :

   - A "Component State" node is connected to the related component node for each component, representing the potential states of components associated with the fault model.

   - A "Global State" node is connected to all "Component State" nodes, managing the global fault distribution (simple/multiple faults) of all "Component State" nodes.

   A graphical representation of the functioning/malfunctioning of a full adder is given in Figure 2.

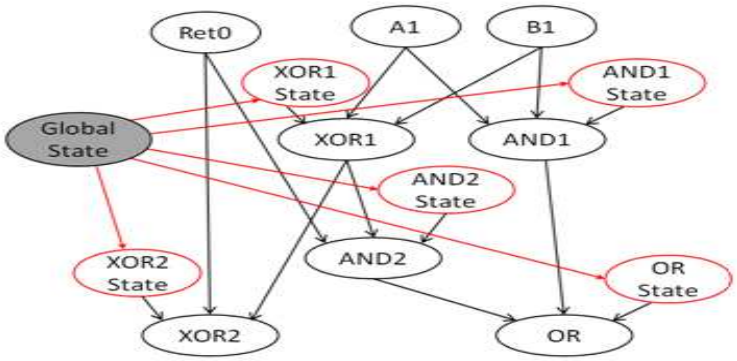



FIGURE 2 – The Bayesian graphical model of a full adder

2. At the "probabilistic level", each node has its own probability table depending on the nodes adjacent to it and the model of faults. Some probability tables of the previous example are shown in Figure 3.

   Circuits could be modeled only with the global state node but this makes the probability tables of components very large. To keep the size of these tables small, component states are used.

### 2.3. A Hierarchical Bayesian network model

The efficient treatment of large circuits is problematic due to the complexity of the inference algorithms. This complexity is generally exponential in the number of nodes of the network [10]. A possible solution to this drawback is the use of a hierarchical representation.
A hierarchical Bayesian network corresponds to a set of Bayesian networks describing the components at a specific level. The lowest level (i.e., the most detailed one) is formed by elementary Bayesian networks consisting of basic components. Figure 4 shows the hierarchical Bayesian network model of a ripple carry adder.
The probability table of each hierarchical node can either have complete information of the internal behavior of the lower level, or a reduced set of information (abstraction). In the first case, the probability tables can be very large and unmanageable, so we give preference to a reduced set of information for the node of higher level which in turn is detailed in the lower one.

| Global State | P(Global State) |
|---|---|
| Ok | 0.95 |
| Bit-flip-XOR1 | 0.01 |
| Bit-flip-AND1 | 0.01 |
| Bit-flip-AND2 | 0.01 |
| Bit-flip-XOR2 | 0.01 |
| Bit-flip-OR | 0.01 |

Global State probability table

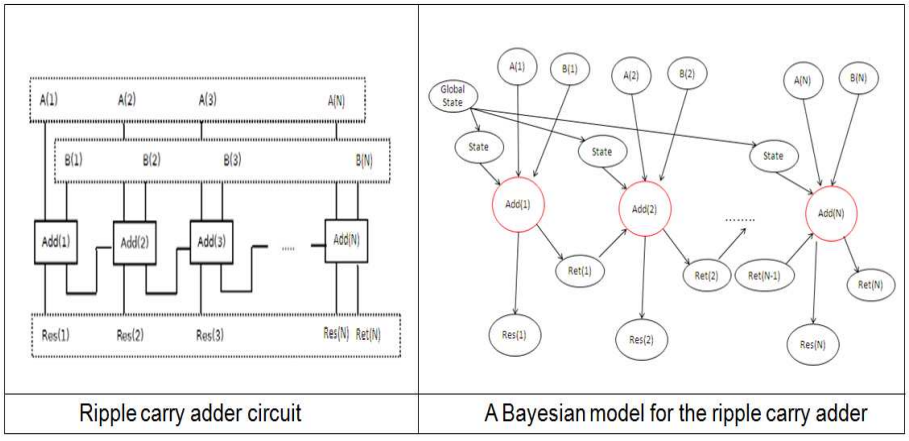
| Global state/ XOR1 State | P(XOR1 State =Ok) | P(XOR1 State=Bit-flip) |
|---|---|---|
| Ok | 1 | 0 |
| Bit-flip-XOR1 | 0 | 1 |
| Bit-flip-AND1 | 1 | 0 |
| Bit-flip-AND2 | 1 | 0 |
| Bit-flip-XOR2 | 1 | 0 |
| Bit-flip-OR | 1 | 0 |

XOR1 State probability table

| A1, B1, XOR1 State/ XOR1 | | | P(XOR1 =0) | P(XOR1 =1) |
|---|---|---|---|---|
| A1=0 | B1=0 | XOR1 State=Ok | 1 | 0 |
| A1=0 | B1=1 | XOR1 State=Ok | 0 | 1 |
| A1=1 | B1=0 | XOR1 State=Ok | 0 | 1 |
| A1=1 | B1=1 | XOR1 State=Ok | 1 | 0 |
| A1=0 | B1=0 | XOR1 State=Bit-flip | 0 | 1 |
| A1=0 | B1=1 | XOR1 State=Bit-flip | 1 | 0 |
| A1=1 | B1=0 | XOR1 State=Bit-flip | 1 | 0 |
| A1=1 | B1=1 | XOR1 State=Bit-flip | 0 | 1 |

XOR1 probability table

FIGURE 3 – Some probability tables for a full adder




Ripple carry adder circuit | A Bayesian model for the ripple carry adder

FIGURE 4 – An example of a hierarchical Bayesian network

## 3. An algorithm for test and diagnosis with optimization

### 3.1. An algorithm for test and diagnosis

In the following, we present an algorithm to determine a set of test vectors allowing the location of all possible faults by means of Bayesian inference.

#### 3.1.1. Principle :

Based on the knowledge of the values injected as input and those observed as the output, the circuit states (global and component ones) can be computed using Bayesian inference. The computed value of each state determines the probability of malfunctioning of each component as represented in the Bayesian network. If one pair of values (input/output) is not sufficient, we select other couples to refine the diagnosis. A final goal is to reduce the number of input values (vectors) to a minimum.

#### 3.1.2. Process :

The algorithm consists of the following steps :

1. Construct the **diagnosis matrix**, whose rows correspond to the input and whose columns to the output values. For each I/O, determine the state of the circuit (Ok or possible faulty components). All possible cases are enumerated.
2. Eliminate redundancy : those vectors (input values) which globally give the same diagnosis information in the matrix are removed (the same set of diagnosis with different or identical output values).
3. According to the observed output values for a selected input vector, the state of the circuit expresses the correct functioning or the malfunctioning of the circuit. If the partial diagnosis corresponds to several potential faulty components, another input vector from the remaining non-chosen vectors is selected to reduce the set of possible faulty components. We continue with the vector selection and the redundancy elimination until we can locate all possible faults.

#### 3.1.3. Example :

We have applied our algorithm to the example of a full adder for the single bit-flip fault model. Figure 5 shows the diagnosis matrix after redundancy elimination ; an entry "-" means that this case is impossible. The different steps of the algorithm are illustrated in Figure 6.

| A1 B1 Ret0/ OR XOR2 | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 000 | Ok | XOR1 XOR2 | AND1 AND2 OR | - |
| 001 | XOR2 | Ok | XOR1 | AND1 AND2 OR |
| 011 | AND2 OR | XOR1 | Ok AND1 | XOR2 |
| 110 | AND1 OR | - | Ok AND2 | XOR1 XOR2 |

FIGURE 5 – The diagnosis matrix for the full adder single bit-flip fault model

#### 3.1.4. Discussion :

As a result, we obtain 4 test vectors instead of the 8 possible ones, the worst case. The selection of the input vectors is crucial for the diagnosis process. In the next section we show how to find a minimum set of these vectors.

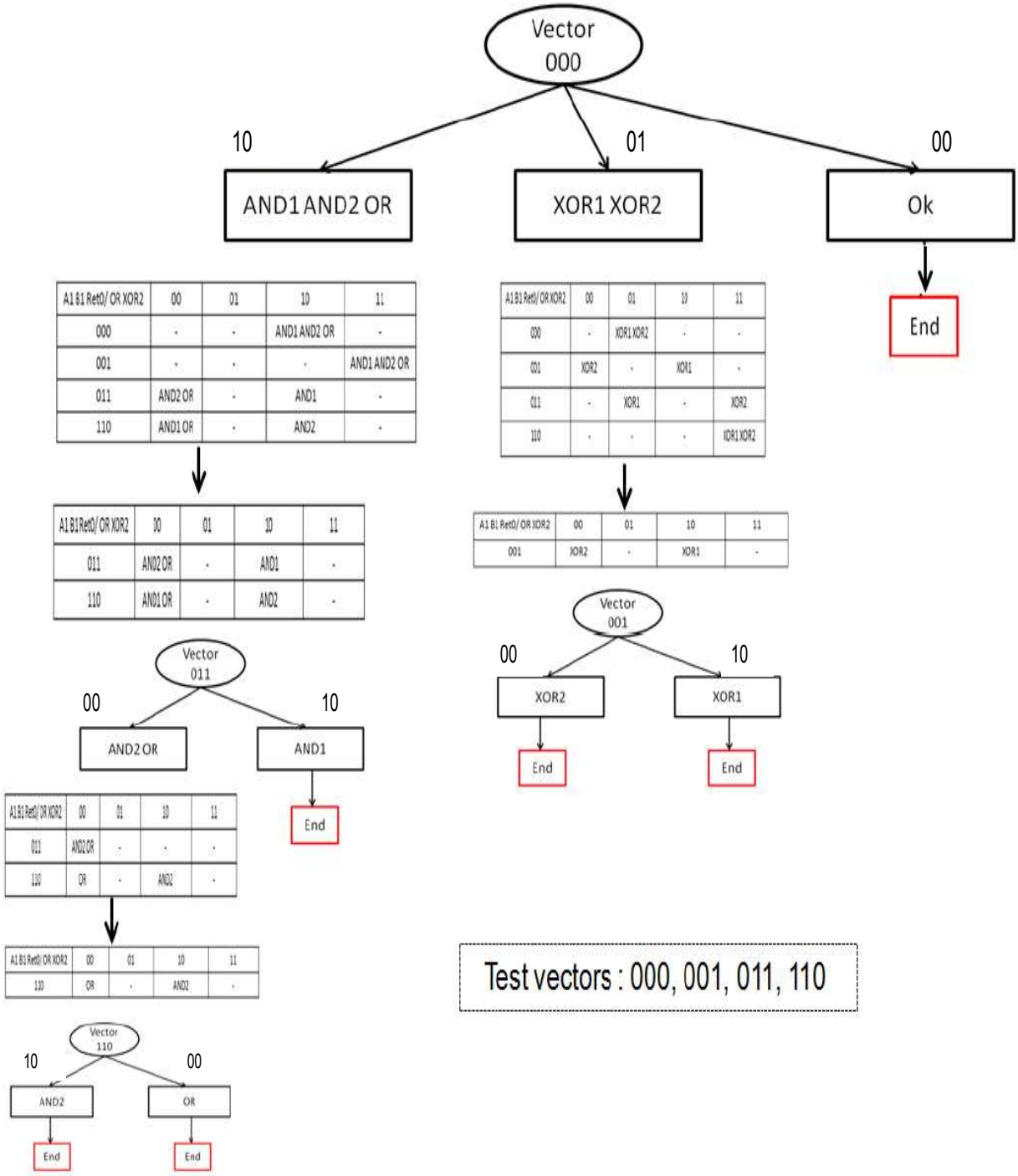



FIGURE 6 – The algorithm applied to the full adder single bit-flip fault model

### 3.2. 3 ways to optimize

Each row of the diagnosis matrix represents an input vector, each column represents the output value and each cell contains a set representing possible states of the circuit (Ok or possible faulty components).

#### 3.2.1. Optimization 1 : first test, then diagnose

The aim of this approach is to determine as early as possible whether the circuit is Ok or not. A reduced set of vectors is used to define the state (Ok or not Ok) of the circuit and gives at the same time a partial diagnosis of faulty components.To refine and to identify the exact faulty component, we apply the previous algorithm.

1. Criteria for the test vector selection

   Let V be a column vector whose rows represent the possible diagnosis (including the Ok case) associated with a set of input/output values, and let $E = \{1, \ldots, Card(V)\}$. Among the vectors that can possibly determine the good functioning of the circuit, we choose those which have in common a minimum of diagnosis cases. We define :

   $$\alpha = Min_{i \in E}\{Max_{j \in E \setminus \{i\}}\{Card(V(j) \cap V(i))\}$$

   We now run the algorithm. In case that the test is not successful to determine the state of the circuit, we select another test in the same way until termination.

2. Example :

   We apply this approach to the full adder with double bit-flip fault model. If we take the diagnosis matrix without redundancy, as shown in Figure 7, we obtain $\alpha = 2$, and we come up with the first test vector (011). At the end, we obtain the 3 test vectors (011), (110) and (001) with a partial location of 13/15= 86,67%.

   If we want to obtain a complete location, the additional vector (111) is necessary.

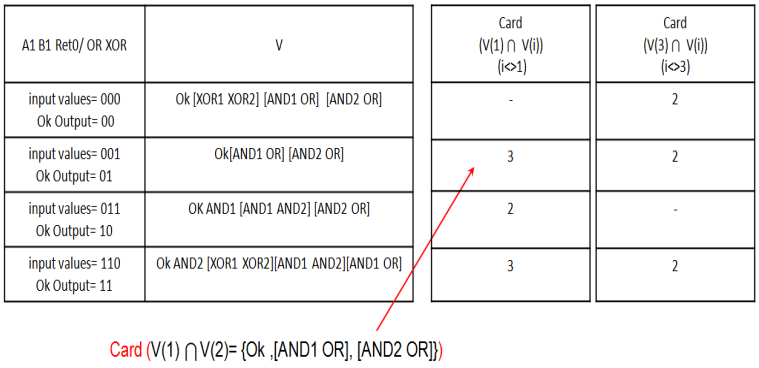



FIGURE 7 – The test vector selection for the full adder considering a double bit-flip fault model

3. Discussion :

   This approach gives a minimum set of vectors for the test but not necessarily a minimum set of such vectors for the diagnosis. In other words, the approach is interesting when we need to test the circuit and obtain some partial information about the location of possible faults.

#### 3.2.2. Optimization 2 : set rules on the choice of vectors

This optimization takes into account the test and the complete diagnosis. We can also run it after the use of the first optimization to locate the remaining faults. The principle is to identify as soon as possible the faulty component (or component group) in the partial diagnosis.

1. Criteria for the selection of a diagnosis vector :
   To identify the faulty component in each cell, we must choose the vector which diagnoses for every case of output values, the smallest number of components (or component groups). Let $E_1$={1, ..., m} and $E_2$={1, ..., n} with m, n representing the number of rows, columns of the diagnosis matrix M, respectively. We define :

$$\beta = Min_{i \in E_1}\{Max_{j \in E_2}\{Card(M(i,j))\}\}$$

   $\beta$ represents the smallest number of faulty components for the partial diagnosis. This value can correspond to the case of one or several vectors.
2. Example :
   If we take matrix M from the previous example without redundancy, see Figure 8, we obtain $\beta = 2$ and the first vector chosen is the one reaching this value 2, i.e., either (011) or (110).

| A1 B1 Ret0/ OR XOR2 | 00 | 01 | 10 | 11 | Max(Card(M(i,j)) |
|---|---|---|---|---|---|
| 000 | Ok | XOR1 XOR2 | AND1 AND2 OR | - | 3 |
| 001 | XOR2 | Ok | XOR1 | AND1 AND2 OR | 3 |
| 011 | AND2 OR | XOR1 | Ok AND1 | XOR2 | 2 |
| 110 | AND1 OR | - | Ok AND2 | XOR1 XOR2 | 2 |

Card(M(2,1))=1 Card(M(1,3))=3

FIGURE 8 – The diagnosis matrix M for the full adder considering a single bit-flip fault model

   The application of this optimization reduces the set of test vectors to 2 (in contrast to 4 vectors obtained previously).
3. Discussion :
   This way to optimize is interesting since it reduces the set of test vectors for the diagnosis and it locates the precise fault. It should be used when the probability of a fault in the circuit increases.

#### 3.2.3. Optimization 3 : Adding evidence

To speed up the diagnosis, we can add evidence due to hardware redundancy which provides certainty of information and hereby reduces the number of cases to be handled. In this approach, we try to find the best evidence for allowing a reduction of the number of vectors.

1. Process :
   When we introduce an evidence, we fix the probability value at 100% Ok or at 100% failed.
2. Example :
   We have tested this approach on the full adder with the single bit-flip fault model. We were able to reduce the number of vectors by adding one evidence (see our results in subsection 3.4).

### 3.3. A hierarchical application

The diagnosis algorithm described in the previous section can be adapted to hierarchical Bayesian networks as follows :
- Apply the algorithm to the highest level.
- Find the faulty component on that level.
- Repeat until the elementary level.

1. Example :

   We give an example of the N-bit adder shown in Figure 4 for the single bit-flip fault model.

   We start by locating the faulty 1-bit adder among all the others, then we apply the algorithm on the elementary circuit (a full adder at the lowest level with single bit-flip fault model). The size of the output vector increases with N and the algorithm's effort rapidly grows from N = 5 on. To face this problem, we first apply the algorithm to the 2- and 3-bit adder and then try to deduce the rest by recurrence. We can show that :

   - For the 2-bit adder, vector (0001) is the test vector locating the faulty 1-bit adder.

   - For the 3-bit adder, vector (000101) is the test vector for locating the faulty 1-bit adder.

   - For the N-bit adder, vector ( 00 $(01)^{N-1}$ ) is the test vector for locating the faulty 1-bit adder.

   3 test vectors are enough to locate the fault in an N-bit adder ; 1 test vector (00 $(01)^{N-1}$) to determine the faulty 1-bit adder (full adder) and 2 to determine the fault within the full adder (as seen previously).

2. Demonstration :

   We remind that the input vector (000) for (A(0)B(0)Ret(0)) gives the output (00) for (Res(1) Ret(1)) if the full adder is Ok, else it gives (01) or (10). The input vector (010) for (A(i)B(i)Ret(i), 1<=i<=N) gives the output (10) for (Res(i+1) Ret(i+1)) if the full adder is Ok, else it gives (11) or (00).

   Suppose that for an N-bit adder, the vector is (00 $(01)^{N-1}$). Then we can demonstrate that for an N+1-bit adder, the vector is (00 $(01)^{N}$).

   We have two cases :

   (a) The fault is not on one of the N first adders :

   We take (00 $(01)^{N-1}$) as input vector for the first N adders. With no fault on the N first adders, this vector gives $(0(1)^{N-1}0)$ as outputs on the N first adders. (i.e., (A(0)B(0)Ret(0)))=(000) gives (00) for (Res(1) Ret(1)) (Ok state for the first full adder), then (A(1)B(1)Ret(1))=(010) gives (10) for (Res(2) Ret(2)) (Ok state for the second full adder), until (A(N-1)B(N-1)Ret(N-1))= (010) which gives (10) for (Res(N) Ret(N)) (Ok state for the Nth full adder).)

   Finally, (A(N) B(N) Ret(N))= (010) can give (10) for (Res(N+1) Ret(N+1)) (Ok state for the N+1st full adder) or (11)/(00) (faulty state for the N+1st full adder). We can conclude the following : if we apply as input (00 $(01)^{N}$) and we have as output $(0(1)^{N}0)$, the (N+1) full adders are Ok. Otherwise, we have as output $(0(1)^{N+1})$ or $(0(1)^{N-1}00)$ and we can say that the N+1st full adder is faulty.

   (b) The fault is on one of the first N adders :

   We take $(00(01)^{N-1})$ as input vector for the first N adders. With a fault on one of the first N adders, this vector determines the faulty adder (hypothesis) and then the N+1st adder must be Ok or it will be an impossible case with 2 faulty adders (context of a single fault).

   Altogether, we can say that $(00(01)^{N})$ is the test vector for the N+1-bit adder.

### 3.4. Results

In this section, we summarize the results of our approach and present a comparative study with a conventional test approach for an adder [11] and a Diagnosis approach using Automatic Test Pattern Generation (DATPG) [5].

1. Comparison with existing work :

   We have applied our approach by taking the hypotheses used in [11]. With the same hypotheses, we have applied the DATPG method [5] on the adders.

   We evaluate the test and the diagnosis coverage. The Test Coverage (TC) or Fault Coverage is defined as the ratio of detected faults to the total number of faults. The Diagnosis Coverage (DC) is defined as the ratio of located faults to the total number of faults.

$$TC = \frac{\text{Number of detected faults}}{\text{Total number of faults}} \qquad DC = \frac{\text{Number of located faults}}{\text{Total number of faults}}$$

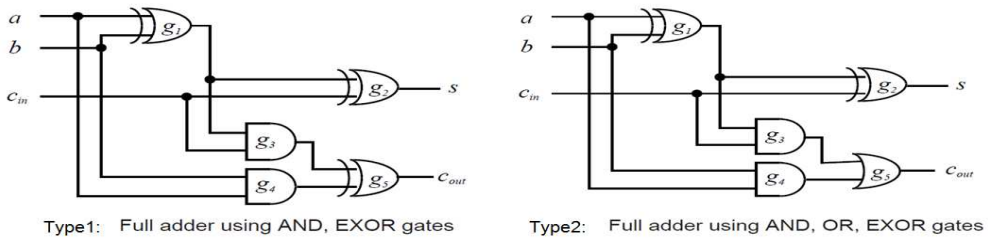



FIGURE 9 – Two full adder types

In [11], a minimum set of test vectors is given to have a 100% TC but the diagnosis is not done. We have taken these vectors and we have computed the DC. In [5], an approach of diagnosis is proposed using the classical ATPG. A 100% TC is given by a minimum set of test vectors that are used for a partial diagnosis and a DC is given. To achieve a 100% DC, extra vectors are necessary and are chosen in an exhaustive way. In our approach, we have applied the combination of optimization 1 and 2 to select the set of vectors properly and we have achieved a 100% TC and a 100%DC.

(a) Hypothesis : we consider two types of architecture for full adders (see Figure 9) together with an N-bit ripple adder (see Figure 4) with stuck-at input/output as fault model.

(b) Results : Table 1 represents the comparative study. The results show that all three approaches give the minimum set of test vectors for a full test coverage, but our approach has the best diagnostic coverage by adding the least number of vectors. We can explain this by the choice and the sequence of vectors. In the two approaches the test and the diagnosis are separated but in our approach we consider test and diagnosis at the same time.

| Full and N-bit ripple adder (Single stuck-at fault model) | Kajihara et al. [11] | DATPG[5] | Our results |
|---|---|---|---|
| Type1 | - 5 test vectors<br>TC= 100%<br>DC= 43.75% | - 5 test vectors<br>TC= 100 %<br>DC= 75%<br>- 7 test vectors<br>TC= 100%<br>DC= 100% | - 5 test vectors<br>TC= 100 %<br>DC= 100% |
| Type2 | - 3 test vectors<br>TC= 100%<br>DC= 31.25% | - 3 test vectors<br>TC= 100%<br>DC= 87.5%<br>- 4 test vectors<br>TC= 100%<br>DC= 100% | - 3 test vectors<br>TC= 100%<br>DC= 87.5%<br>- 4 test vectors<br>TC= 100%<br>DC= 100% |

TABLE 1 – Test and diagnosis for adders (TC : Test Coverage, DC : Diagnosis coverage)

2. Diagnosis exploration :

In this part, we are interested in evaluating the impact of the proposed optimization on the set of test/diagnosis vectors. For this, we consider the full adder of type 2 (see Figure 9) with bit-flip or stuck-at input/output as fault model. Type 2 is the most efficient as seen in the first part of our results.

Table 2 shows that the algorithm without optimization gives a complete localization, and it even reduces the number of test vectors, but the number of these is not minimum. Using the optimizations reduces this number more in the bit-flip fault model because we have less faults. Using

optimizations 2 and 3, we can retain the complete localization. This approach is more useful as the number of faulty circuits increases. therwise optimization 1 is the most suitable.

The proposed algorithm reduces the number of test vectors for diagnosis. The proposed optimizations minimize this number depending on the fault model and the need. Optimization 1 is suitable for a test with partial localization and optimizations 2 and 3 for complete localization. In the case of complex circuits, the best solution is to mix these optimizations. First, test the circuit using optimization 1. Then, if the coverage is not enough, use optimization 2 to improve the coverage, or pass to a complete localization.

| Full adder type2 | Algorithm | Optimization 1 | Optimization 2 | Optimization 3 |
|---|---|---|---|---|
| Single bit-flip | - 4 test vectors<br>Complete diagnosis | - 1 test vector<br>Partial diagnosis | - 2 test vectors<br>Partial diagnosis | (Evidences XOR1)<br>- 3 test vectors<br>Complete diagnosis |
| Single stuck-at | - 5 test vectors<br>Complete diagnosis | - 3 test vectors<br>Partial diagnosis | - 4 test vectors<br>Complete diagnosis | (Evidences XOR1)<br>- 3 test vectors<br>Complete diagnosis |
| Multiple bit-flip | - 5 test vectors<br>Complete diagnosis | - 3 test vectors<br>Partial diagnosis | - 5 test vectors<br>Complete diagnosis | (Evidences XOR1)<br>- 4 test vectors<br>Complete diagnosis |

TABLE 2 – Full adder result

## 4. Conclusion and perspectives

In this paper, we have proposed a fault diagnosis approach based on Bayesian networks. We have shown that we can introduce any fault model. The Bayesian network inference allowed to find for each input/output the sets of potential faults and this is easily done with appropriate Bayesian tools. From that, we have proposed an algorithm whose objective is to reduce the set of vectors for a complete or a partial fault location. A complete study was done on adder examples demonstrating that the diagnosis based on Bayesian networks together with different optimizations gives a minimum set of test vectors in case of complete or partial diagnosis.
As perspectives, we would like to apply our algorithm to other regular circuits and evaluate the performance for transient faults. We are also interested in online diagnosis with more efficient schemes of inference computation.

## References

[1] M. Abramovici, M. Breuer, and A. Friedman, "Digital systems testing and testable design," *Wiley-IEEE Press*, 1990.

[2] I. Pomeranz and S. Reddy, "On dictionary-based fault location in digital logic circuits," *IEEE Transactions on Computers*, vol. 46, pp. 48–59, 1997.

[3] M. Amyeen, D. Nayak, and S. Venkataraman, "Improving precision using mixed-level fault diagnosis," *Proceedings of the IEEE International Test Conference (ITC '06)*, pp. 1–10, 2006.

[4] Y.-C. Lin, F. Lu, and K.-T. Cheng, "Multiple-fault diagnosis based on adaptive diagnostic test pattern generation," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 932–942, 2007.

[5] Y. Zhang and V. Agrawal, "A diagnostic test generation system," in *Test Conference (ITC), 2010 IEEE International*, pp. 1–9, Nov 2010.

[6] D. Cao, S. Kan, and Y. Sun, "Design of reliable system based on dynamic bayesian networks and genetic algorithm," in *Proceedings of the Reliability and Maintainability Symposium (RAMS)*, pp. 1–6, 2012.

[7] T. Rejimon, K. Lingasubramanian, and S. Bhanja, "Probabilistic error modeling for nano-domain logic circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 55–65, 2009.

[8] B. Benware, C. Schuermyer, M. Sharma, and T. Herrmann, "Determining a failure root cause distribution from a population of layout-aware scan diagnosis results," *IEEE Design & Test of Computers*, vol. 29, no. 1, pp. 8–18, 2012.

[9] P. Naim, P.-H. Wuillemin, P. Leray, O. Pourret, and A. Becker, *Réseaux bayésiens*. 2007.

[10] V. Delcroix, S. Piechowiak, and J. Rodriguez, "Diagnostic à base de réseaux bayésiens hérarchiques," *National Days of Models Reasoning, INRETS ESTAS*, pp. 51–66, 2001.

[11] S. Kajihara and T. Sasao, "On the adders with minimum tests," in *Proceedings of the Sixth Asian Test Symposium (ATS'97)*, pp. 10–15, 1997.