FOURTEENTH EUROPEAN ROTORCRAFT FORUM

Paper No. 38

MODE SYNCHRONIZATION ALGORITHM

FOR ASYNCHRONOUS AUTOPILOT

A. SILVA

AGUSTA SISTEMI
TRADATE (VA), ITALY

20-23 September, 1988
MILANO, ITALY

ASSOCIAZIONE INDUSTRIE AEROSPAZIALI
ASSOCIAZIONE ITALIANA DI AERONAUTICA ED ASTRONAUTICA

MODE SYNCHRONIZATION ALGORITHM FOR ASYNCHRONOUS AUTOPILOT

Ing. Antonio Silva
AGUSTA SISTEMI S.p.A
Via Isonzo 33 21049 TRADATE ITALY

ABSTRACT:

Asynchronous Architectures with dissimilar redundancy can be employed in Fault-tolerant applications to improve system's insensitivity to common-mode failures - due to flaws in H/W or S/W - caused by the temporal patterns of the inputs from the outside world.

A problem related to asynchronous Architectures is the synchronization of all changes in the system's operational modes, caused by discrete events that are sampled by each computational node asynchronously with respect to the other nodes and to the input transition itself.

The proposed algorithm, based on a concept of "Retrospective Agreement" (agreement among all the valid nodes on the recognition of a pattern of a given duration in the input stream) achieves the best compromise between fast response and noise and phase insensitivity, the latter being complete in terms of "all or none" mode change, and flexible to accommodate redundancy management and variable noise filtering effect.
The only assumptions are that all nodes in the system have the same basic frame cycle and are connected to each other for limited information exchange.

The need for total agreement could prevent synchronization from occurring if one of the nodes fails only with respect to a particular piece of data, no failures being detected by Built-in Test. The algorithm, though, can be applied to the disagreement condition, achieving synchronized isolation of the faulty node and subsequent reconfiguration.

True independent operation in the computational nodes may then be kept, taking full advantage of the Asynchronous Architecture, while ensuring positive and quasi-synchronized mode transitions throughout the whole system.

INTRODUCTION

Modern technology for Guidance Systems in Aircraft and Missiles is nowadays trying to meet the ever-growing demand for performance, together with extensive Fault-tolerance and operational safety characteristics, particularly in Helicopter Stability Augmentation Systems with high authority .

Distributed computing permits to achieve considera ble performances, allowing to implement multiple redundancies, as well as cross-monitoring and related fault management policies, resulting in a high degree of Fault-tolerance.

The use of multiple redundancy has brought forward the major issues of Common Mode Failures, capable to defeat the System.
The availability of several microprocessor families with comparable computing power gives now the opportunity to incorporate dissimilarity in the redundancy and monitoring scheme.

Common Mode Failures not covered by dissimilarity are those caused by particular input data values, yielding wrong results, due to flaws in the Hardware or in Software. Since any computing system is basically a sampled-data system, this may happen in all nodes if all nodes are sampling data exactly at the same instant, i.e. if the system is synchronous.

To overcome this, Asynchronous Distributed Architectures have been developed, where the nodes are deliberately out of phase by a random percentage of the basic cycle rate (frame) that remains however the same for all nodes (isochronysm) to keep the system to a manageable complexity.
This approach makes the chances that the flaw shows up in all nodes negligible, and the System Design is forced towards loosely-coupled Architectures, effective in fault-propagation prevention, characterized by the absence of critical common Hardware, in charge of providing all nodes with synchronizing signals.

The use of distributed Architectures brings in problems, the most obvious of which, and certainly not the least significant is that the processors, or , more generally, the computing nodes, will have to communicate with each other to permit both synergy and cross-checking.
It is actually essential for the System to ensure that different nodes do not generate contrasting commands or messages or take decisions in opposite directions.

SYNCHRONISATION (or Equalisation): the Analogue Case

Synchronization (or consolidation , or equalization) of a single analogue datum starting from signals coming from multiple asynchronous sources is a well known topic.
Several techniques, such as Median Selection, plain or weighted Averaging, ARIMA processes or combinations of the above have been developed , are continuously being refined, and are widely used to calculate in each asynchronous node a single datum that is sufficiently similar to that calculated in any other node.

These techniques can be easily made (or are intrinsically) insensitive to the failure of any single node , incorporating faulty node datum isolation, and in general guarantee that the output datum , for each node, is constantly drifting, more or less smoothly, towards a datum that is continuously calculated , according to the technique used, as an appropriate "average" of all valid input signals; this implies they are intrinsically robust and safe.

Furthermore, in multiple sampled-data systems, the difference in the sampled values in any node pair cannot exceed an upper bound, rigorously determined by the signal's slew rate and the worst-case delay in sampling from one node to another. In general, though, this upper bound is significant when the system is sampling data with a basic cycle that is so slow to be comparable with Shannon's Theorem's minimum rate; in practical systems a basic cycle faster than that at least by an order of magnitude is the rule.

SYNCHRONIZATION (or Consolidation): The Discrete Case

Treating signals with boolean values obviously prevents the use of techniques such as averaging or filters. Actually, considering the signals in continuous time, the techniques used for analogue signals are capable to converge, if the signals show a steady state of sufficient duration, but the algorithm output, with a final threshold to obtain again a boolean, yields unacceptable transient differences, causing a subset of nodes to consolidate a boolean opposite to the other nodes.

Another difficulty is that, treating booleans coming from asynchronous sources in the boolean domain, no upper bound can be established that limits the difference in output from one node to the others. The instantaneous difference in the output from any two nodes can be either 0% or 100%, and the only type of error upper limit that can be determined is in terms of phase and duration of the output inequality.

Inequality can be caused in fact by various mechanisms:

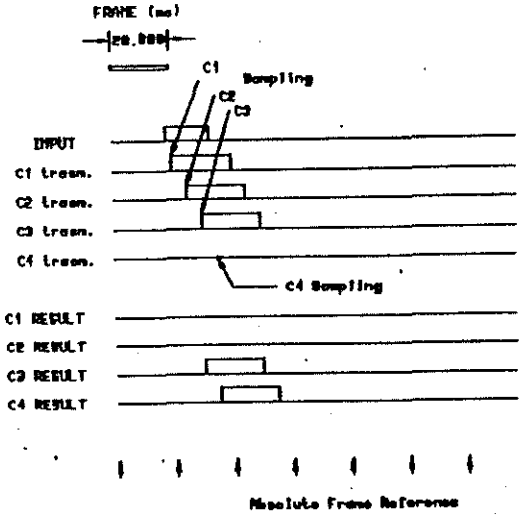
- The sampling process applied to discretes causes the following:
  - Pulses shorter than a full basic cycle can remain undetected in a subset of the nodes.
  - Every sampled level is stretched in duration to an integer number of basic cycles (frames).

- Different phase in sampling causes the following:
  - Sampling of opposite levels on transition boundary
  - Internode communications appear with additional delay at each node
  - Any input pulse may be measured with a 1-frame duration difference in any node pair.
  - Voting and threshold mechanisms of any kind fail to consolidate consistently in all nodes.

A typical example showing voting techniques inadequacy for this case follows:

Consider four nodes, C1..C4 , with a basic frame cycle τ , where C1 is synchronized with an absolute frame of period τ, while C2,C3,C4 are out of phase by 1,2 and 3 times $\tau/4$, respectively. Let us assume that the algorithm is defined as:

If 3 out of 4 among the available signal sources agree, the output is the agreed level, otherwise it remains unchanged from the previous frame.
The 4 sources are the direct input and the signals read by the other three nodes, and it is assumed that the time from sampling the input and broadcast to the other nodes is short with respect to the frame.



It becomes evident that if a pulse of duration $1/2\tau<\Delta<3/4\tau$ starts just before the start of the absolute frame, it will be directly sampled by C1,C2 and C3, but not by C4. But C1..C3 transmit their sample to each other and to C4 before C4 samples its inputs. The result is that C1 and C2 will not satisfy the algorithm condition for agreement, while C3 and C4 will, even if C4, for instance, takes its decision in contrast with its own sampled value .

Threshold techniques fail as well, since they are basically voting on the event of a certain time being elapsed with certain conditions, and time measurements can differ by one frame, causing the threshold to be exceeded only in a subset of the nodes in the asynchronous case.

SYNCHRONIZATION OF MODES: The nature of the Problem

The major problem in multinodal systems is that of synchronization of operational mode changes in consequence of events of discrete nature: external switches, reaching of thresholds, timeouts ,etc. In synchronous systems , typically equipped with a "data input interrupt", all nodes are sampling data at the same instant, and thus differences may only be due to malfunctions. Transmission of samples from one node to others is synchronized as well, and the transmission delay is a known constant, in absolute terms. Appropriate algorithms guarantee discrete signals consolidation in a minimum number of frames, simultaneously in all nodes.

In Asynchronous multinodal systems this holds no longer. The problem changes from the selection of an algorithm that consolidates the output in the minimum number of frames to the use of an algorithm that allows consolidation at all.

In most such systems, discrete events are in fact the key data on which decisions are made about the operational mode the whole system should assume. Examples are Manoeuvre Mode, Attitude Hold, Trim, Trim Synchronization, Automatic transitions, and so on. Note how a mode change of this sort causes in most cases a re-initialization of long-term datums in the system's Control Laws, and from this point of view a transition has long memory, or "latches"; in other cases a true latching process is effectively required.
This means that, for each node consolidating a transition, the following processing is altered.

This makes a Synchronized Consolidation of the mode transition compulsory, putting time response at a lower priority. The need is in fact to ensure that the whole system changes its operational mode in the same way, or does not change mode at all. In other words, the priority is to achieve system's consistency at all times.

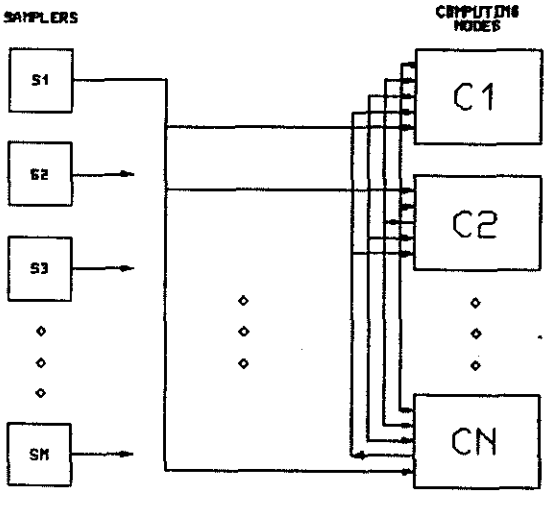
Systems with multiple cross-checking redundancies are particularly critical in this respect, since even temporary inconsistencies leading to different operational modes in different nodes can cause the checks to detect failures (that actually do not exist) that will generate cut-out commands. This occurs for certain when the decision result is latched in a subset of the nodes.

To overcome this, the only alternative to a proper synchronizing algorithm is to decrease the cross-checking sensitivity, causing a reduction in error detection time response and redundancy management performances, that is not acceptable in most cases.

A TYPICAL ASYNCHRONOUS ARCHITECTURE

Let us now consider a typical architecture with multiple asynchronous sampling and computing nodes.

Consider N computing nodes, C1..CN, able to communicate to each other and receiving data from the outside world via M sampling nodes, S1..SM, able to provide fresh data to C1..CN with a frequency higher than the basic frame τ of the System (equal in period for all (C) nodes).
Let us assume that (S) refresh the data to (C) in "broadcast" mode, i.e. that data is available to any node in (C) at the same instant. Communications between the nodes are assumed "broadcast" as well, and each node is able to transmit data also to itself. Internodal communication must be such to guarantee complete transfer of relevant data from Ci to Cj in much less than a frame time τ.



Each Ci will freeze, at the beginning of each of its frames, and for the entire frame duration, the whole set of input data, consisting of :

- data from (S)
- data from the other nodes
- data from itself

and the "staleness" of any of these data is $\leq \tau$.

Note how, from the point of view of the computing nodes (C), the input data can be considered simply a set of external asynchronous data, and as such the presence of the samplers (S) will be ignored in the following.

THE ARCHITECTURE MODEL AND THE ALGORITHM REQUIREMENTS

Since each Ci is a sampler with sampling rate τ, the phase relation between Ci and Cj is the only relevant figure. It is always possible, without loss of generality, to rename the members of {C} so that Ci is the node receiving first the new value of a certain piece of data, and all other nodes C2..CN sample the same data in sequence, with CN sampling with a delay Δ with respect to Ci, where $0 <= \Delta < \tau$.

Note ,however, that it is important to consider the timing of internodal communication, since these repeat with a period τ and thus have an effect at the sampling rate. The isochronism of such communication and the sampling in all nodes , even if the transmission bursts are out of phase, is the key to a simple solution to the synchronization problem.

The worst case of asynchronism may, in virtue of the possibility to renumber the nodes, be simulated by an even temporal distribution of nodes over the period τ, with the first node synchronous with an absolute sampler of period τ, without losing generality, since sampling makes a phase delay of τ/N identical to any other phase delay in the range $[0..\tau)$.

The case of two or more nodes in synchronism is simpler than the case above, and is anyway covered by the proposed method .

The analysis of the problem with multiple trials in several directions and bearing in mind the constraints imposed by the real-time nature of the systems, has generated the following requirements for a mode synchronization algorithm:

- independent operation: the algorithm must be the same in all nodes and must be executed independently at the same point in each node's basic cycle τ.

- insensitivity to noise: the algorithm must guarantee a synchronized response in the whole system , even in the presence of fast and random transients in the inputs.

- quasi-synchronization on output: all the nodes must synchronize the transition with a maximum difference between each other shorter than τ.

- robustness: it must be impossible for any one node to ignore a transition synchronized by other nodes.

- reconfigurability: the algorithm must be easily extendable to include features like error detection and faulty node masking.

- efficiency: the communications load between nodes, being the system strictly connected, must be reduced to the minimum. Additionally, since the algorithm must be executed every frame in all nodes, it must be computationally efficient, and must allow a substantial parallelism in inputs and outputs.

- time response: the basic algorithm must have the minimum time response to a signal going to a steady state condition that ensure the requirements above to be met. It must, however, be easily extendable to filter out transients shorter than an arbitrary duration.

DESCRIPTION OF THE ALGORITHM

The proposed algorithm is based on a most intuitive consideration:

If the System is composed of asynchronous samplers and none of the nodes knows its phase relationship with the others, the single node is not able to conclude about an agreement among all nodes, in any selected instant .

If however we consider all the discrete signals coming from all nodes in the system as continuous signals in the time reference of a single node, and a time window is examined with a sufficient duration (integer number of frames) in the immediate past, every node is able to compare the pattern of a signal, for all nodes, as it has evolved in time.

The comparison between all the temporal images of the same signal as seen by all the nodes allows to define a concept of "retrospective agreement", that means: if an agreement cannot be sure for the current sample, it may be possible in the past, if the patterns in the past are close enough for a sufficient duration.

The algorithm is more precisely defined as follows:

Given that:

1 All nodes are sampling all inputs and execute the algorithm every frame of duration τ

2 Each node communicates to all nodes in the system (including itself) the value of the direct inputs as sampled by it within a time $<<\tau$ from the sampling instant.

3 Every node applies the algorithm to the images coming from the other nodes and from itself (the latter read during the previous frame (see note)).

Then the result is calculated as follows:

* Chosen a window of N frames, if the patterns received from *ALL* nodes show a common level of the signal in at least N-1 consecutive frames, then the result is made equal to the common level.

* If none of the two logic levels satisfies this condition, the logic value of the result remains equal to the one at the previous frame.

NOTE : This last clause is not mandatory , since a difference of a full frame is inessential, but is useful to make the following discussion simpler.

From the clauses 1 and 2 we can say that:

a) the maximum duration difference for any of the transmitted signals, as sampled by a node reading the communication data, is limited to one frame, source to source, since the isochronism of the samplers limits the differences only to phase effects.

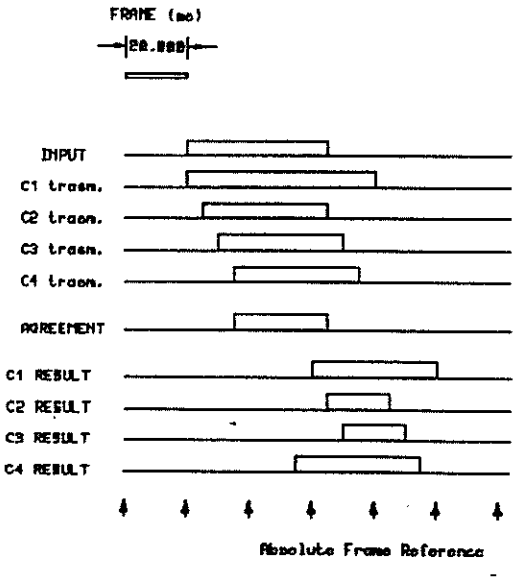
b) for the same reasons, the maximum difference, in terms of temporal position of the patterns, is one frame.

The algorithm will then certainly consider agreed all signals with a steady state duration $> (N-1)* \tau$, and will behave as an absolute filter for all signals with steady state duration $< (N-2)* \tau$.
For signals with a steady state duration within the limits above,the result depends on the phase relation between the nodes, but yields the same result in

all nodes, independently, with a difference in the result phase and duration between any two nodes limited to one frame.
The full agreement required by the algorithm automatically limits the pattern steady-state duration to the minimum recognized in any one node.
Relative to the signals from the other sources, this minimum signal is either in phase or out of phase by one frame, and anyway always in the same sense, since otherwise the worst case phase shift between any two nodes would be 2 frames, contradicting b).

The fact that the patterns have to be recognized within a time window that is one frame longer than the pattern target length accommodates for both duration and phase differences , and guarantees that the event of recognizing the agreement happens (with a tolerance of one frame in absolute time) in all nodes.
The differences between results among the nodes in terms of resulting pulse duration and phase can be easily explained in an absolute time reference.
The algorithm has in fact been conceived in absolute terms, since the signal relations are more evident and remain undistorted by the sampling in a particular node.
Let us consider a simple example consisting of 4 completely asynchronous nodes, C1..C4 , equally distributed , in phase, on the frame τ. (Note that this is also a worst case for synchronization).
The signals retransmitted by each node after sampling are shown. The effect of sampling is evident, stretching the transmitted signals to integer multiples of τ. The effect of phase relations is also evident, both on sampled values and on the phase of retransmission.

In continuous terms, it appears clear how the conditions a) and b) are satisfied, but interesting is the virtual signal that identifies , in the continuous time, the full agreement between all four nodes. This signal is the temporal inter section of the signals transmitted by all nodes, and as such is unique (in the continuous time) and is shorter than or equal to any of the tran smitted signals.



If this signal comprises N-1 contiguous sampling instants for any node, then ALL nodes will recog nize an N-1 frames pattern in all signals in an N-frame window.
In fact the full agreement signal starts synchro nously with the sample of the latest node , C4 , and , to contain N-1 samples for it, must be of duration greater than $(N-2)*\tau$.
But, since this signal is the temporal intersection of all transmitted signals, this means that *ALL* these signals have a duration greater than $(N-2)*\tau$ and , since the signals come from sampling, their duration cannot be but an integer multiple of τ, and so must be at least $(N-1)*\tau$.

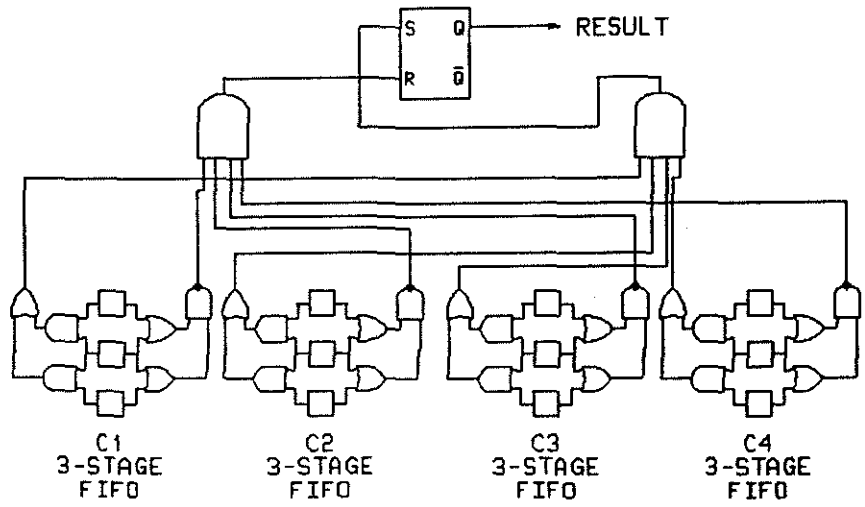
This implies that at least N-1 contiguous samples are present in all the patterns, and , since the phase relation is limited within one frame, the patterns are recognized within an N-frame window in *ALL* nodes.

On the contrary, if no node exists for which the full agreement signal comprises N-1 samples, this means that at least for one node the transmitted signal is shorter than N-1 frames, i.e, N-2 frames or less, and no node will be able to recognize an N-1 frame pattern in all signals within the window.

This technique can be applied to both logic levels independently, and for N = 3 the mutual exclusion of agreement on the two levels is intrinsically guaranteed, and so no ambiguities are possible.

When no agreement can be reached on either level, i.e. during fast transitions of the input, or in response to short pulses, the algorithm does not change the result from that obtained at the previous frame. This decision may be considered questionable, but, also in the light of examples, and considering that in a sampled data system short pulses may be lost due to its nature anyway, seems the most sensible approach; obviously, the result will have to be properly initialized at startup, and the signals history as well, to obtain a consistent steady state.

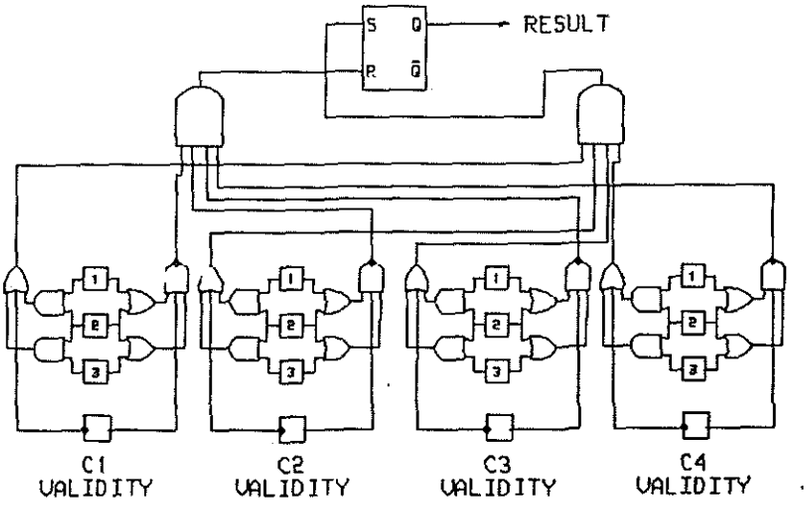
The following logic network depicts this algorithm for N = 3 , working on both logic levels, with 4 asynchronous nodes.



Note that result retention in case of no agreement has been implemented in the simplest way, tying the two agreement signals to the inputs of a S/R Flip-Flop. Since the two agreement signals are mutually exclusive and the lack of agreement does not alter the Flip-Flop status, its output Q is the desired result.

FAULT MASKING EXTENSION

In a redundant system, it is common practice to implement complex Built-in Test functions, capable to isolate malfunctioning nodes. The algorithm can be extended to incorporate a "masking" of faulty nodes' data, simply forcing the pattern recognition signals for such nodes to their active state, before the final full agreement gate, as shown in the figure.



The agreement is then dependent only on the signals coming from the working nodes, since the agreement of faulty nodes is forced unconditionally.

ERROR DETECTION AND ISOLATION CAPABILITIES

This algorithm can be extended to include two forms of error detection and management: faulty node data isolation and failsafe operation.

- Faulty node data isolation.

Apparently, this algorithm , although safe, can lead to a stuck system only because a single failed node remains in disagreement with the others.
This condition, however, if the actual input discretes are not in continued transition, can be synchronized as any other boolean among all nodes, or at least among all working nodes.
If the failure of a node is such to deeply alter its functions, the Built-in Test functions will isolate that node within reasonable time. If instead the failure is more subtle, like reading a single discrete in a stuck level in that node, the agreement among all nodes (possibly excluding the suspect node), on the fact that that node has been in disagreement with all others for a sufficient period of time, guarantees that the set of working nodes synchronously decides to exclude it, and, if the suspect node is able to run the algorithm, it is able to decide, independently and synchronously, to shut itself down.

- Failsafe operation.

In the case when common mode errors affect all nodes of one type, and no majority agreement is reachable, it is however always possible to reach an agreement on an unresolved situation timed out, and on this basis to force the result to a failsafe level in all working nodes. This condition must latch and appropriate messages must be sent to the outside world.

EFFICIENCY IN CASE OF MULTIPLE DISCRETE INPUTS

Since the algorithm is based only on logical operations on bits stored in memory, and since the majority of processing units is able to operate simultaneously on all the bits in an arithmetic unit's register, if the history of signals belonging to the same class, i.e. with same N-frame window, is kept in memory with all bits "packed" in words , it is possible to apply the algorithm on the entire word, obtaining a word result and reducing execution time by a factor equal to the number of discretes processed in parallel.
The masking operations may be done in parallel , as well as fault isolation and reversion to failsafe result.

CONCLUSIONS

Asynchronous redundant architectures have long been regarded as risky, in terms of possible artifacts occurring due to their nature and difficult to anticipate, in particular in management of boolean entities. This algorithm is believed to relieve much of these concerns and to allow such architectures to fully exploit their capabilities, with a reasonable price in terms of computational loading.
Asynchronous loosely coupled architectures, including simpler and cheaper hardware configuration, no hardware weak points, independent processing, lower chance of common mode failures and overall benefits in terms of cost and reliability , can be employed without the deterrent of extensive communication needs to solve synchronization issues.