Fast Zone Discrimination

Robin Clark[*]
R&D Department
Energy Technology Control
25 North Street, Lewes, BN7 2PE, Great Britain

Abstract

This paper describes an algorithm (Fast Zone Discrimination - FZD) for analysing Concrete Euler diagrams and listing all present zones.

One application area of Euler/Spider diagrams, is modelling failure modes in electronic circuits. For this a procedure is required to check Spider diagrams for logical consistency by ensuring that all present zones in a model have been considered. Unused zones could be viewed as un-handled failure conditions. In order to know which zones have not been used, a complete list of present zones is required for each concrete diagram drawn.

To determine if zones are present, a concrete diagram must be examined using programmatic area operations. These area operations are costly of computer time and it is desirable to eliminate all that are unnecessary.

The algorithm initially builds two sets of relationships from a concrete diagram and then uses these to target searches for zones that may be present. Using the two sets of relationships eliminates checking for a large number of missing zones; thus processing diagrams quickly and efficiently.

Keywords: Euler, Fast Zone Discrimination, present, available region, java area, algorithm.

1 Introduction

Definition 1.1 *An Euler diagram is a finite set of of simple closed curves in the plane.*

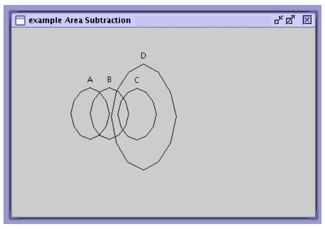
In earlier work[1] spider diagrams [2] have been used to represent the failure modes of components and modules within safety critical electronic systems. By using logical reduction and hierarchies of abstraction, mathematical modelling of complete safety critical systems is possible.

Spider diagrams are based on Euler diagrams. This paper looks specifically at determining which zones are present in an Euler diagram.

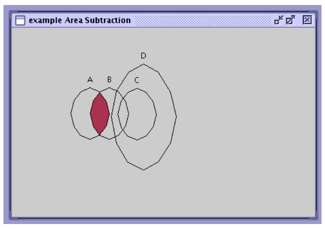
A zone is defined as a region in the plane formed by the intersection of curves in the set I (the ‘included’, or inside set) and a set of curves E, representing the curves excluded from the set. For instance in figure1(a) there is a zone described by $I = \{B, C, D\}$ with $E = \{A\}$.

One way of looking for present zones would be to look for every possible combination and then to use the excluded zones to check for obscuration. Checking all possible combinations is henceforth referred to as the ‘Brute force method’.

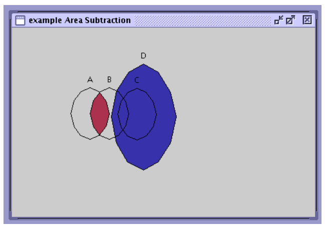
[*] `r.clark@energytechnologycontrol.com`



(a) Euler Diagram



(b) Intersection A B



(c) area to subtract

Figure 1: Simple Euler Diagram

The brute force method is simple, and would be practical for small numbers of contours. However as constraint/spider diagrams become used in practice larger numbers of contours and very large diagrams will become common. The Brute force method for finding present zones is of the complexity order $NC.2^{NC}$ (where NC is the number of contours in the Euler diagram). Because of this, a more efficient algorithm has been sought.

In order to obtain information from the concrete diagram, a Java class called an Area[3] is used. This provides Area operations such as *exclusive or* and *subtract*. To measure the effectiveness of a zone searching algorithm, the number of Area operations is considered an appropriate metric.

Section 2 defines determining ‘present’ zones, and provides Euler diagram examples showing checking for the existence and obscuration of zones. Section 3 defines relations between contours in an Euler diagram, and introduces the terms ‘pure intersection’, ‘pure intersection chain’ and ‘enclosure’. The mathematical properties of these relationships are then discussed and defined. Section 4 shows how these relationships can be used to draw graphs representing Euler diagrams, and discusses a practical algorithm implemented from spanning these graphs. Section 5 compares the performance of the algorithm with the ‘brute force’ method.

2 Determining Missing and Present Zones

A ‘present’ zone is simply one on which one can place an object (or spider ‘foot’). For instance there may be an area of intersection on a diagram that is obscured by other contours. That intersection is impossible to use and therefore not considered ‘present’ in the diagram.

Actually determining whether a zone is present or not in a simple diagram is easy by inspection. An example follows describing two zones and how they are proved ‘present’ or otherwise, using ‘Area’[3] operations.

2.1 Zone Present Example

Let us examine an intersection and determine whether or not it is 'present' in the diagram. In figure 1(a) the intersection $A \cap B$ can be observed. To define this intersection we can say that it has a set of included contours $I_n = \{A, B\}$ and a set of excluded Contours $E_n = \{C, D\}$ (where n is the nth zone under investigation).

By looking at the area formed (figure 1(b))we can see that the intersection exists. The area of all other contours in the diagram must now be subtracted from the intersection to check for obscuration (see figure 1(c)). After subtraction of the areas formed by $C \cup D$ there is an area remaining of the intersection $A \cap B$.

The Zone $A \cap B$ is therefore present in this diagram.

Definition 2.1 (Obscuration) *A zone Z formed from the intersection of contours $c_1...c_j$ is said to be obscured by a collection of contours $o_1...o_k$ when*

$$\bigcap_{i\in 1...j} Interior(c_i) \subset \bigcup_{i\in 1...k} Interior(o_i)$$

2.2 Zone Missing Example

Consider the potential zone $C \cap B$. This would have an included set $I_N = \{B, C\}$ and an excluded set $E_N = \{A, D\}$. It can be seen that subtracting the interior formed by $A \cup D$ from the interor formed by $B \cap C$ as an area operation; leaves nothing. The zone $B \cap C$ is therefore considered missing in this diagram.

2.3 The General Case : Proving a Zone is Present

The total number of contours in the diagram, will be referred to as NC.

For some diagram elements the contours will not interact and therefore searching for zones can be applied within subsets of contours. These subsets are defined later in the paper. The variable, Interacting contours count, ICC_n represents the number of contours within these subsets. It will always be less than or equal to the number of contours in the diagram.

$$ICC_n \leq NC \quad (1)$$

A zone can be defined by two disjoint contour sets, included I_n and excluded E_n, where n is the zone under investigation.

First of all we determine the concrete area formed by the intersection set I_n. This involves taking the intersections of all the interiors of the sets in I_n, as area operations.

$$AreaIntersection_n = \bigcap_{x\in I_n} Interior(x) \quad (2)$$

The result of these intersection area operations could be a NULL area, indicating that there is no intersection. If the intersection does exist we then need to ensure that it is not covered up by any of the contour interiors formed by the set E_n.

In order to check for obscuration,we must find the area formed by all other sets that could cover it. This is done by taking the union of all the excluded contours, as an area operation.

$$AreaExclusion_n = \bigcup_{x\in E_n} Interior(x) \quad (3)$$

The $AreaIntersection_n$ may now have the $AreaExclusion_n$ subtracted from it, as an area operation. This is an area subtraction and area is only subtracted where it overlaps. If the result has non zero area, then the zone is considered present.

$$\begin{gathered} RemainingIntersection_n = \\ AreaIntersection_n - AreaExclusion_n \end{gathered} \tag{4}$$

Finally the sets I_n and E_n must contain all contours for the the Euler diagram under investigation.

$$InteractingContours = I_n \cup E_n \tag{5}$$

For reference a table of all possible zones, showing which are present in the four contour diagram (figure 1(a)), is shown below.

Inside	*outside*	
Included set I	*Excluded $set E$*	*Present*
{ }	{ D C B A }	Y
{ A }	{ D C B }	Y
{ B }	{ D C A }	Y
{ B A}	{ D C }	Y
{ C }	{ D B A }	N
{ C A }	{ D B }	N
{ C B }	{ D A }	N
{ C B A }	{ D }	N
{ D }	{ C B A }	Y
{ D A }	{ C B }	N
{ D B }	{ C A }	Y
{ D B A}	{ C }	N
{ D C }	{ B A }	Y
{ D C A }	{ B }	N
{ D C B }	{ A }	Y
{ D C B A }	{ }	N

3 Relationships that can be obtained from an Euler Diagram

Intersection areas in the concrete diagram can be formed in two ways. By enclosing another contour, or by overlapping a part of another contour. These two types of intersection are clearly mutually exclusive.

The intersections that do not involve enclosure have been termed 'pure intersections'.

The algorithm developed in this paper applies two initial searches to the diagram. The first looks for enclosure relationships and the second for pure intersections. These searches are applied to the cross products of all contours in the diagram.

Definition 3.1 (Enclosure) *we say that contour A encloses contour B if* $Interior(A) \supset Interior(B)$

Enclosure for a given pair of contours is expressed in the boolean equation 6.

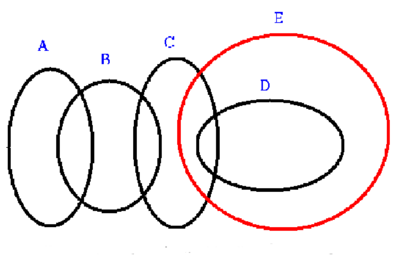
$$enc(a, b) = \ (Interior(a) \supset Interior(b)) \tag{6}$$

Definition 3.2 (Pure Intersection) *we say that there is a pure intersection of a, b, where there is an intersection $a \cap b$ but a does not enclose b, and b does not enclose a.*

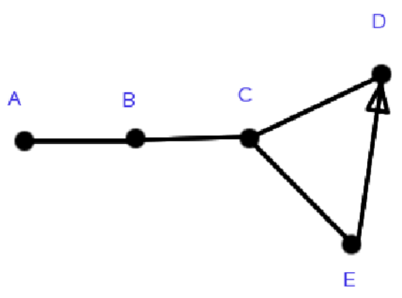
Pure intersection for a given pair of contours is expressed in the boolean equation 7 using the definition of enclosure above.

$$pi(a, b) = ((Interior(a) \cap Interior(b) \neq \emptyset)) \wedge \neg enc(a, b) \wedge \neg enc(b, a) \wedge a \neq b \tag{7}$$

Because the definition of pure intersection expressly forbids enclosure, we have



(a) Euler Diagram



(b) Graph of diagram

Figure 2: Pure Intersection Chain with Enclosure

Lemma 3.1 (Mutually exclusive) *Pure intersection and Enclosure are mutually exclusive.*

3.1 Relationship Properties

By applying the equations 6 and 7 to the cross product of all contours in the concrete diagram, we have two lists of relationships, the pure intersections and the enclosures.

From the diagram in figure 2(a), we obtain the following relationships.

3.1.1 Pure intersections relations

Pure Intersection relationships for the diagram in figure 2(a) are :-

$$A \xrightarrow{pi} B, B \xrightarrow{pi} A, B \xrightarrow{pi} C, C \xrightarrow{pi} B, D \xrightarrow{pi} C, C \xrightarrow{pi} D, E \xrightarrow{pi} C, C \xrightarrow{pi} E$$

3.1.2 Enclosure Relations

Relationships for the diagram in figure 2(a) include one enclosure within a pure intersection chain.

$$E \xrightarrow{enc} D$$

Examining these relations, we can classify them.

3.2 Transitive

For pure intersection relationships, it does not follow that (looking at figure 2(a)) because A purely intersects with B, and B with C that A purely intersects with C.

$$A \xrightarrow{pi} B \wedge B \xrightarrow{pi} C \nRightarrow A \xrightarrow{pi} C$$

.

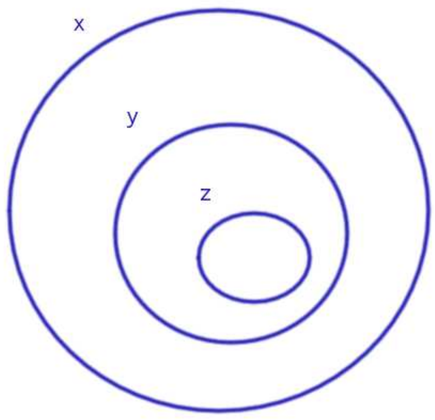


Figure 3: Enclosure within Enclosure : A transitive relationship

Pure Intersection is therefore not a transitive relationship.

Figure 3 demonstrates the transitive nature of enclosure relationships. For the contours in the diagram $\{x, y, z\}$ it can be seen that because x encloses y, and y encloses z, x encloses z. Enclosure is based on a relationship of proper subsets of areas, and is therefore a transitive relationship.

$$x \xrightarrow{enc} y \ \wedge \ y \xrightarrow{enc} z \ \Rightarrow \ x \xrightarrow{enc} z \tag{8}$$

3.3 Reflexive

A reflexive relationship is one where an element can be related to itself [4]. Clearly enclosure and pure intersection are anti-reflexive (i.e. no pure intersection or enclosure can be reflexive) because they require interactions between contours.

3.4 Symmetric

A symmetric relation is one such that $a \rightarrow b \Rightarrow b \rightarrow a$ [4]. Clearly enclosure cannot be symmetric. Because pure intersection is defined by sharing an intersection (see eqn 7, i.e.without enclosure), pure intersection relations are always symmetric. The relationships defined for the pure intersections above (3.1.1), can now be expressed thus.

$$A \overset{pi}{\leftrightarrow} B, B \overset{pi}{\leftrightarrow} C, D \overset{pi}{\leftrightarrow} C, C \overset{pi}{\leftrightarrow} E$$

3.5 Pure Intersection Relationship Properties

Pure intersection relationships are

- Not Reflexive
- Symmetric

Note that by following pure intersection relationships, sets of contours connected by pure intersections can be determined. These are referred to as 'pure intersection chains'.

Contours in the pure intersection chain may obscure other members in the same chain (see figure 2(a)). Formally a 'pure intersection chain' is defined thus

Definition 3.3 (Pure Intersection Chain) *Let d be an Euler diagram : a pure intersection chain is a maximal set of contours C in d such that for any pair of contours in C there exists a sequence of contours such that*

$$c_i \xrightarrow{pi} c_n \; for \; i = 1, ..., n-1$$

3.6 Enclosure Relationship Properties

Enclosure relationships are

- Not Reflexive
- Not Symmetric
- Transitive

4 Methods for finding Present Zones

Firstly we can consider the diagram in terms of pure intersection relationships. The effects of enclosure and obscuration are dealt with later.

Lemma 4.1 (pure intersection cases) *When analysing an Euler diagram from pure intersection relations, there are only three possible cases that can be presented. Lone contours, lone purely intersected contours and pure intersection chains.*

4.1 3 cases for zone identification

The three cases are described in greater detail below.

4.1.1 Lone Contours

Definition 4.1 (Lone Contour) *A Lone Contour is a countour not intersected by any other.*

A lone contour will always represent one present zone, which may include enclosing contours (if any). The contours in figure 3, are all lone contours, and analysing these determines the following present zones.

Lemma 4.2 (Lone Contour Single Zone) *A lone contour will always produce one present zone in an Euler diagram.*

lone contour under investigation	*Included Set* I_n	*Excluded Set* E_n
$\{z\}$	$\{xyz\}$	$\{ \}$
$\{y\}$	$\{xz\}$	$\{y\}$
$\{x\}$	$\{x\}$	$\{yz\}$

4.1.2 Lone Pure Intersection Pairs

Definition 4.2 (Lone Pure Intersection Pair) *A Lone Pure Intersection is a pair of intersecting contours, not belonging to a pure intersection chain.*

The pure intersections in figure 6(a) are all lone pure intersections.

Lone pure intersections may be enclosed by other contours (see eqn 8). Three present zones will always be found upon analysing a lone pure intersection. Enclosing contours (if any) are added to the intersection sets I_n.

Lemma 4.3 (Lone Pure Intersection 3 zones) *A lone pure intersection will always produce three present zones in an Euler diagram.*

4.1.3 Pure Intersection Chains

Pure intersections can contain zones formed by multiple intersections, they can have contours within the chain that obscure zones, and they may contain enclosure within the chain.

In describing the process for finding the present zones within a pure intersection chain it helps to graph them, with the contours becoming vertices, enclosures forming directed edges and pure intersections forming undirected edges.

Circuits of undirected edges indicate the possibility of multiple intersections within the chain. The graphs and their meanings are dealt with in the next section.

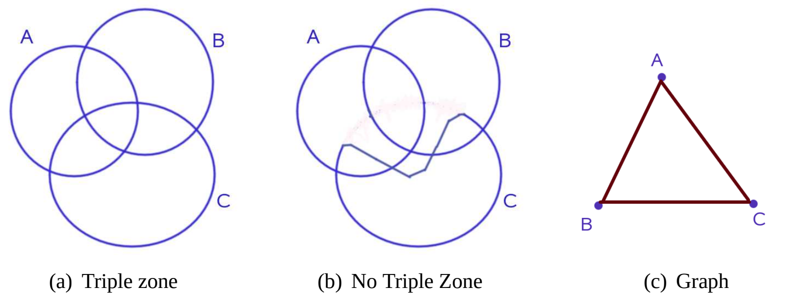
Pure intersection chains within a diagram can be viewed as separate groups within the Euler diagram. That is to say, they may be analysed in isolation with the enclosure rules being applied afterward. This means that a diagram can be broken down into smaller more manageable chunks and this significantly reduces the number of area operations to perform (see eqn 1). The reasons for this are described in the section 4.4. (i.e. area operations need only be performed within pure intersection chains). Thus the number of contours in the diagram NC, comprises of smaller sets of contours (of size ICC_n) that can be analysed separately. See eqn 1.

Contours may enclose pure intersection chains. Where this is the case all enclosing contours (see eqn 8) are added to the intersection sets I_n of all zones discovered within the pure intersection chain.

4.2 Graphing Pure Intersection and Enclosure Relationships

By representing contours as vertices, enclosure relationships as directed edges (because they are transitive) and pure intersections as undirected edges (because they are symmetric), the diagram in figure 2(a) can be represented by the graph in figure 2(b).

4.3 Graphs with Circuits



(a) Triple zone (b) No Triple Zone (c) Graph

Figure 4: Graph with Circuit

Ignoring enclosure for the time being, consider the Euler diagrams in figures 4(a) and 4(b) .

These both produce the same graph representation see figure 4(c).

Note that the existence of a circuit of undirected edges (i.e. pure intersections) indicates the possibility of a multiple intersection. If the area operations in equations 2, 3, 4 and 5 are satisfied the multiple area is ’present’.

Note the corollary of this, *if no circuit exists then no multiple intersection can*. The principle strength of the algorithm hinges on this. By targeting the searching to only zones that ‘may’ be present, all those that cannot are by-passed.

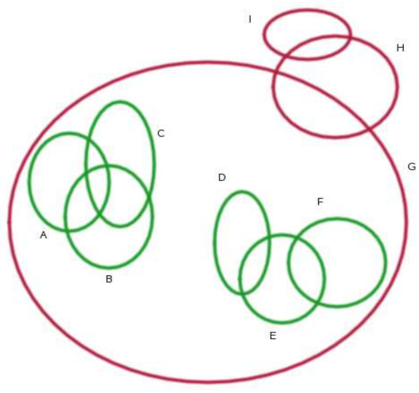
Note that the circuits within the pure intersection chain can be investigated, and then the enclosures can be added afterward. This is because ‘pure intersection’ and ‘enclosure’ are mutually exclusive. Also the transitive relationship established for enclosure, means that we simply have to add enclosure intersections to the included set I_N (see eqn 8).

Lemma 4.4 (Venn N circuit) *For a Venn N zone to exist there must be a circuit in the corresponding graph of pure intersections with N vertices, where each vertex corresponds to a contour in the Venn N intersection.*

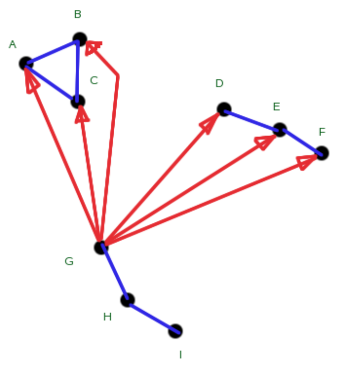
Lemma 4.5 (Venn N pure intersection chain) *For a circuit of pure intersections to exist, they must be all members of the same pure intersection chain.*

4.4 Non connected graphs

Consider the Euler diagram in figure 5(a).



(a) Separate Pure Intersection Chain



(b) Graph of Separate pure int.chains

Figure 5: Non-Connected Graphs

There are three separate pure intersection chains in this diagram. They are $\{A, B, C\}$, $\{D, E, F\}$ and $\{G, H, I\}$. Note that present zones found from the pure intersection chains $\{A, B, C\}$ and $\{D, E, F\}$ all include an intersection with contour G.

By finding present zones within the pure intersection chains, and adding any enclosing contours (see eqn 8) to the intersection set I_N the present zones germane to the pure intersection chain are discovered.

A more complicated scenario is when enclosure occurs within a pure intersection chain, see figure 2(a). In analysing the Intersection $C \cap D$ it will not be recognised as an present zone, because the enclosure relationship $E \xrightarrow{enc} D$ will be applied and the intersection becomes $C \cap D \cap E$: after passing obscuration testing (area subtraction of $A \cup B$), the zone with $I_N = \{C, D, E\}$ and $E_N = \{A, B\}$ will be registered as a 'present'

4.5 Algorithm Design

The aim of this algorithm is to avoid the burdensome 2^{NC} complexity order of checking for all possible zones in an Euler diagram with NC contours.

By breaking the diagram into a number of smaller sets of contours, which can be checked in isolation, the number of checks is significantly reduced.

The smallest possible sets that can be analysed in isolation are the 'lone contour', the 'lone pure intersection' and the 'pure intersection chain'. A 'lone contour' will always produce one present zone (see lemma 4.2). A 'lone pure intersection' will always produce 3 (see lemma 4.3).

A 'pure intersection chain' can potentially produce 2^{ICC_n} present zones (where ICC_n is the number of contours in the chain).

One could check for all 2^{ICC_n} possible zones within the 'pure intersection chain'. However, the 'pure intersection chain' is handled more effeciently than this by only applying checks to circuits of pure intersections within the chain (see lemma 4.4).

By applying the enclosure relations to the present zones discovered in each of the three cases, all present zones in the diagram are discovered.

The correctness of the algorithm rests on the lemmas 3.1, 4.1, 4.2, 4.3 and 4.4.

4.6 Algorithm Pseudo Code

In high level pseudo code, the algorithm works thus:

```
BEGIN
 Determine all Enclosure relationships.
 Determine all Pure intersections
 Search through pure intersection relationships and obtain pure
 intersection chains.

 For all contours in the diagram where contour is not a member
    of a pure intersection chain; add all enclosing contours;
    register as an present zone;

 For all pure intersections in the diagram where pure
    intersection is not a member of a pure intersection chain;
    add all enclosing contours to intersections register as an
    present zone;

 For all pure intersection chains
   for each pair intersection within chain add any enclosing
   contours and determine if the intersection is present using
   area operations; if present then; register as a present zone;
   obtain all circuits within the pure intersection chain;
   for all circuits within the pure intersection chain;
      add any enclosing contours and check obscuration using
      area operations;
      if the zone indicated is present; then register as an
      present zone;
END
```

4.7 Checking for all Possible zones - Brute force/Binary Count

The number of potential zones in an Euler diagram containing NC contours is 2^{NC}.

In a diagram with NC contours, each zone under investigation will comprise of the set I_n and the set E_n. The number of area operations to get the intersection area, and the number to get the obscuration check area, will always add up to NC, using the 'brute force' method.

For instance were a diagram to contain 32 contours, to brute force check for the existence of all contours would take all possible combinations of 32 objects. This corresponds to a binary count and thus 2^{32} possible zones to check for. A diagram with 32 contours would contain a potential of over 4 billion zones. Multiply that by the 32 area operations (with varying proportions of intersection (I_n) and obscuration (E_n) tests - but always adding up to 32) required and we reach an astronomical number (32.2^{32}).

In general then the brute force zone search, with intersection area operations, and obscuration testing (on average $\frac{NC}{2} + \frac{NC}{2}$), takes

$$NC.2^{NC} \tag{9}$$

4.8 Number of Area Operations using Pure intersection and Enclosure Relationships

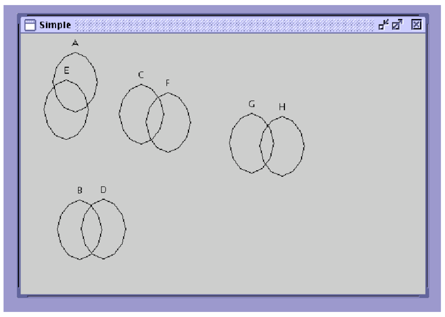
The number of area operations applied by this algorithm depends upon the complexity of the diagram, and the sizes of the pure intersection chains. Were the worst case to apply, i.e.$Venn_{NC}$ this algorithm is less efficient by $2.NC^2$ i.e. for a Venn N diagram, the number of area compares for the FZD algorithm is

$$2.NC^2 + NC.2^{NC} \tag{10}$$

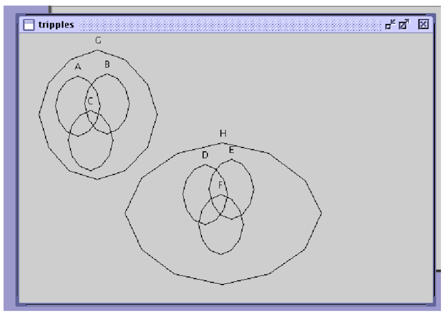
Most diagrams written by human beings will be far less complicated than $VennN$. In the domain of safety critical circuit/system analysis, the diagrams will be comprised typically of a number of separate pure intersection chains, and the searches need only be applied within them. The ICC_N value for interacting contours will be equal to the number of contours in each pure intersection chain. Also because the graphs are traversed, *most contour combinations will be determined impossible by the fact that no circuit exists in the undirected edges*.

5 Case Studies

To compare the algorithms performance against the 'brute force' method, I have taken two diagrams of a typical complexity level that is drawn from the failure mode[1]



(a) Simple Pure Intersection Pairs



(b) Two Enclosed Venn 3

Figure 6: Performance Comparison

applications.

The number of area operations necessary to find all zones has been analysed and a numerical formula based on NC, has been derived using techniques from [5]. This formula can now be plotted to compare the performance of the algorithm for the two test patterns. The examples shown in figures 6(a) and 6(b), use 8 contours per diagram. Using the Brute force method these would require $NC.2^{NC}$ or 2048 area compares to determine all visible zones. By duplicating the structures, values can be calculated for general NC number contour diagrams of the same family. The algorithm parses the relations built in the first two passes to eliminate unnecessary searches. These relationships are held in Java data structures in RAM and are therefore considered to have minimal impact on processing time. For this reason, only the java Area[3] operations are considered in comparing the performance of the algorithm against the 'brute force' method.

5.1 Simple pairs of contours

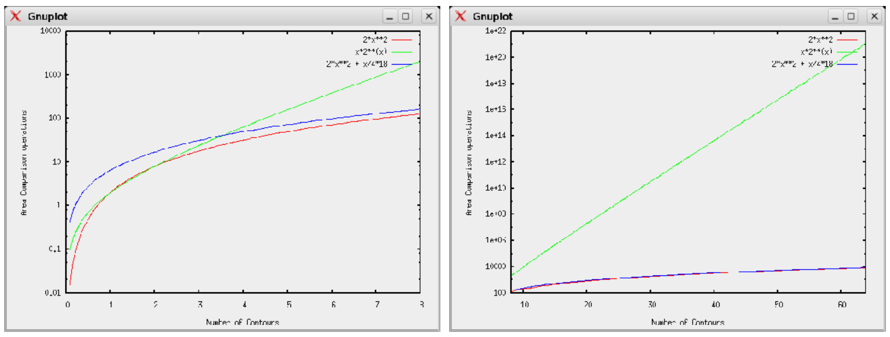
The simple diagram, shown in figure 6(a), consists of four overlapping pairs of contours. To determine the enclosure and pure intersection relations, two cross products of contour area searches are required. Thus 2 NC^2, i.e. 128 searches. Zones derived from lone contours and lone pure intersections do not need to be checked for existence or obscuration. The total number of area compares/operations is therefore $2.64 = 128$. Were one to add more lone pure intersections to this diagram, the diagram would become larger, but would have the same pattern. Five lone pure intersections would take $2.128 = 256$ area operations to find all present zones.

As a general case, for extrapolating larger diagrams of the same pattern, where N is the number of contours

$$AreaOperationsRequired = 2.NC^2 \tag{11}$$

5.2 Two Venn 3 totally Enclosed Once : a more Complex Diagram

The second diagram, see figure 6(b), contains two Venn 3 configurations each enclosed by a contour. Breaking this down, we have two single zones (from the contours G and H). Examining the two Venn3 structures, these require an existence check for the triple intersection (3 area operations). As the number of contours to check for obscuration



(a) 0 to 8 contours

(b) 8 to 64 contours

Figure 7: Performance Comparison

against it is 0, they do not require obscuration testing. Within each Venn3 each of the double intersection zones must be checked for obscuration. Thus 2 area operations to construct the shape of the zone, and 1 area operation to test for obscuration, thus 3 per pair. The three single zones in the pure intersection require 1 area operation to construct the shape of the contour, and two to test for obscuration. Thus 3 per pair. This diagram therefore requires $128 + 2.(9 + 9) = 146$ area compares. As a general equation for the number of the number of area operations required can be calculated, thus:

$$AreaOperationsRequired = 2.NC^2 + \frac{NC}{4}.(18) \tag{12}$$

5.3 Extrapolating for N Contour Diagrams

Duplicating the structures in the diagrams in figures 6(b) and 6(a), and using the general case equations (11 and 12), a plot of area searches required against diagram complexity can be drawn.

These graphs were produced in Gnuplot[6] (which uses a Fortran [7] like syntax for formulas), with the following equations:

$GnuplotSyntax$	$LineColour$	$Arithmetic$
$x * 2 * *x$	$Green$	$NC.2^{NC}$
$2 * x * *2 + x/4 * 18$	$Blue$	$2.NC^2 + \frac{NC}{4}.18$
$2 * x * *2$	Red	$2.NC^2$

These graphs clearly shows that the FZD method efficiency increases with the number of contours in a diagram.

6 Conclusion

6.1 Practical Implementation

An algorithm has been implemented in Java, which finds present zones in an efficient and quick way, in a spider diagram editor application. It has been checked against a ‘brute force’ algorithm, by inspection, with Venn4, Venn5 and a variety of test diagrams.

6.2 Future Enhancements

Because Surface Areas are calculated as a side effect of the Java[3] area class, some well formed-ness[8] criteria can be checked for.

Further efficiency may be possible by analysing the structure of the graphs produced from the pure intersection chains, and determining rules to further reduce the number of Java area operations to prove a zone present.

References

[1] R. Clark. Failure mode modular de-composition using spider diagrams. In *Proceedings of Euler Diagrams 2004*, volume 134 of *Electronic Notes in Theoretical Computer Science*, pages 19–31, 2005.

[2] J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. *LMS Journal of Computation and Mathematics*, 8:145–194, 2005.

[3] Sun Micro Systems. Java area operations. Available from http://java.sun.com/j2se/1.3/docs/api/java/awt/geom/Area.html, 2000.

[4] David Tall Ian Stewart. *The Foundations of Mathematics : ISBN 0-19-853165-6*. Oxford University Press, 1977.

[5] Gregory J.E. Rawlins. *Compared to What ? An introduction to the analysis of algorithms ISBN 0-7167-8243-x*. Computer Science Press, 1991.

[6] Various Open source Project. Gnuplot 4 home page. Available from http://www.gnuplot.info/, 2005.

[7] A. Balfour D.H. Marwick. *Programming in Standard Fortran 77 ISBN 0-435-77486-7*. Heinemann Educational Books, 1979.

[8] Gem Stapleton Peter Rodgers, John Howse. Properties of euler diagrams. http://www.cmis.bton.ac.uk/research/vmg/papers/, 2007.