An Analysis of U.S. Patent #5,243,538 "Comparison and Verification System for Logic Circuits and Method Thereof,"

Randal E. Bryant
Carnegie Mellon University

August 30, 1994

Abstract

On Sept. 7, 1993, U. S. Patent Number 5,243,538 was awarded to four Hitachi employees for a technique for comparing logic circuits using Binary Decision Diagrams (BDDs). Although the U. S. patent application was filed on Aug. 7, 1990, they had filed for a patent in Japan on Aug. 9, 1989, and hence this earlier date should be used in determining the prior art.

The key property exploited by the patentees is that by generating the BDDs for the two logic circuits according to a unique ordering of the variables, the task of comparing the BDDs is greatly simplified. The patentees did not cite and apparently were not aware of work done on BDD-based logic comparison during the 9 years between the cited papers by S. B. Akers and their patent application. In particular, the author devised a technique for generating and comparing logic functions using Ordered (i.e., assuming a unique ordering of variables) BDDs in 1984. Papers published in 1985 (*Design Automation Conference*) and 1986 (*IEEE Transactions on Computers*) describe this method in detail. We assert that this patent should not have been issued, as it attempts to patent prior art.

Keywords: logic verification, binary decision diagrams, software patents

1. Summary

On Sept. 7, 1993, U. S. Patent Number 5,243,538 was awarded to four Hitachi employees for a technique to compare logic circuits using Binary Decision Diagrams (BDDs). Although the U. S. patent application was filed on Aug. 7, 1990, they had filed for a patent in Japan on Aug. 9, 1989, and hence this earlier date should be used in determining the prior art.

The patent cites ten other patents and four technical papers, with papers [Akers78] and [Akers80] having the greatest relevance. The Hitachi approach uses the BDD generation algorithm presented in [Akers78] to generate BDDs for the two circuits to be tested for equivalence. It then uses a comparison algorithm that differs from that of [Akers80]. The key property exploited by the patentees is that by generating the BDDs for the two logic circuits according to a unique ordering of the variables, the task of comparing the BDDs is greatly simplified.

The specific implementation described in the patent is not correct. There are cases where it will fail to detect that two BDDs represent the same function. This error could be corrected by applying one additional transformation rule during the BDD comparison step.

The patentees did not cite and apparently were not aware of work done on BDD-based logic comparison during the 9 years between Akers' papers and their patent application. In particular, I devised a technique for generating and comparing logic functions using Ordered (i.e., assuming a unique ordering of variables) BDDs in 1984. Papers [Bryant85] and [Bryant86] describe this method in detail and were widely available well before the patent application in Japan. The approach reported in these two papers is very similar to that of the patent. In the concluding remarks of [Bryant86], a variant is described that even more closely matches the Hitachi approach. My description does not contain the error of the one described in the patent. Other researchers had constructed systems with all of the essential features of the Hitachi system, describing them in papers published over one year before the Japanese patent was filed [Madre88].

The potential economic value of this patent is significant. Several commercial programs, including ones by IBM, Synopsys, and CAD Language Systems, allow the user to compare two versions of a logic circuit using BDDs. Many tools developed in universities and for internal use by companies also use these techniques. Given the publications on the subject predating the patent application, Hitachi would have a hard time arguing both that their invention is novel and that these programs infringe upon it. To date, Hitachi apparently has not attempted to gain from this patent. Searching a database of U. S. patents issued up to June, 1994, I can find no other patents making use of BDDs, nor any others on logic verification. Undoubtably more will be issued in the future.

This paper examines the Hitachi approach in depth, comparing their method to the earlier work by Akers, as well as the more recent, uncited work. It is argued that the patent should not be held valid, as every aspect of the patent claimed to be novel is in fact covered in [Bryant86]. The paper concludes with a discussion of how they obtained a patent for an idea that had been published several years earlier. There is no evidence of deliberate deception on the part of the patentees. It appears simply that neither they nor the patent examiner were aware of the more recent technical literature relevant to the subject of the patent.

2. A Note on Presentation Style

Scientists and engineers familiar with the technical literature in digital circuit analysis will find the presentation style of the patent rather peculiar. First, it is clear that the technical content was generated by people with a limited command of English. As an example, the abstract begins with the statement:

> "When a hierarchy design is attempted in a logic design of a logic circuit, a system for verifying an equivalence between an upper level logic and a lower level logic is required." [Patent, Abstract]

A more conventional rendering of this sentence would be:

> "When a logic circuit is designed hierarchically, a system for verifying equivalence between the representations at higher and lower levels of abstraction is required."

Although the reader can extract the main ideas despite the deficiencies in the English presentation, some of the details are hard to follow. Nonetheless, I believe my interpretation of the technical content is reliable.

Second, the language used in a patent is quite different from that used by scientists and engineers when communicating technical results to their peers. Due to controversies about the patentability issues of software, the patent never uses terms like "algorithms" or "programs." Instead, the approach is described alternatively as a "system," to qualify it as an apparatus, or a "method," to qualify it as a process [PTO89, p. 6] [OTA93, p. 132]. The need to describe the approach in such forms leads to some rather awkward contrivances. For example, one component of the claimed system is:

> "... a controller controlling a start and stop of said Boolean expression extractor, said BDD production device, said BDD simplification device and said comparator." [Patent, Claim 1]

Such a controller is, of course, realized by the computer program guiding the overall verification process. Words like "said" and "thereof" are also not commonly found in technical writing.

As a final contrast, the standards of scholarship for a patent are much different from those of a technical paper. There is no attempt at establishing the correctness of the approach. In fact, the approach as presented contains an error. There is no attempt to demonstrate the approach is practical through either theoretical analysis or experimental results. The only statement about performance is:

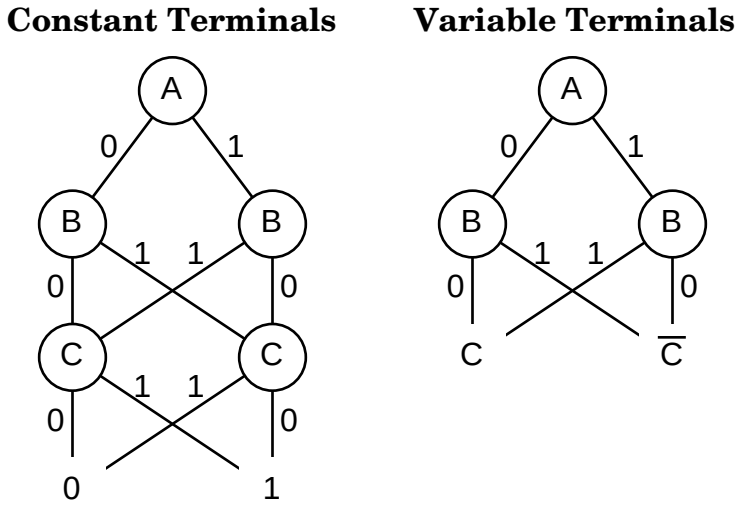

Figure 1: BDD representations of odd parity function

> "Furthermore, according to the verification method of the present invention, the comparison and judgement is carried out at a high speed. For example, when the number of the variables is 20, the speed in the prior art is K, but the speed of the present invention is improved ten times over that of the prior art: K/10." [Patent, Column 10]

Such a statement would never be accepted by the technical community. First, it is not clear to what "prior art" they are comparing their method. Second, the performance of a logic comparison program typically depends on many more parameters than just the number of variables. It is highly unlikely that the Hitachi approach obtains a uniform speed-up of 10 for all 20-input circuits.

For the remainder of this discussion, I will attempt to avoid judging the patent as if it were a technical paper. Instead I will focus on the actual ideas embodied in the patent and how they relate to prior work.

3. Introduction to BDDs

A binary decision diagram (BDD) represents a Boolean function by a form of data structure known as a "directed acyclic graph." Note that the use of the term "graph" in this context is not related to the kind of graph with which one displays data. This graph consists of a set of "nodes" (also called "vertices") where each node is labeled by an input variable to the function. Each node has two outgoing "branches" labeled '0' and '1' corresponding the the two possible values of the node variable.

Figure 1 illustrates BDD representations of the odd parity function over input variables A, B, and C. This function is defined as follows. Counting the number of

inputs set to 1 gives a number in the range 0–3. The function yields 1 if this count is odd and 0 if it is even. The figure shows two variants on the BDD representation. In the form on the left hand side, used in [Bryant86], the bottom-most branches lead to constant values 0 or 1. For a given set of input values, the function value is obtained by following a path from the top (the "root") to one of these constants. At each node, the path takes the branch corresponding to the value assigned to the node variable.

In the form on the right hand side, used in [Akers78] and [Patent], a branch can lead to a variable (e.g., C) or to the logical complement of the variable (e.g., $\overline{C}$). If the path followed during evaluation leads to such a label, the function value equals the value assigned to the variable or its complement. The choice between these two variants is of minor significance when implementing BDD-based logic comparison. Allowing the more general terminal values can reduce the BDD size somewhat, at the cost of slightly more complex processing.

4. Overview of Hitachi Approach

We will organize our discussion of the patent according to its first claim [Patent, Column 11]. We will discuss the significance of the other 9 claims later. Claim 1 describes the invention as a "system" as follows:

> "A system for comparing and verifying an equivalence between two logic circuits . . . "

It defines 6 parts to this system:

Part 1. Storage of circuit descriptions

> ". . . a storage device storing a first logic corresponding to a first of said two logic circuits and a second logic corresponding to a second of said two logic circuits, said first and second logics expressed by a logic circuit form, a truth table form or a Boolean expression form. . . . "

This is simply standard language to cast the approach as an "apparatus," one of the forms of patentable inventions [PTO89, p. 6]. Note that the term "logic" here means the description of a combinational logic circuit.

Part 2. Boolean expression generation

> ". . . a Boolean expression extractor converting said stored first and second logics to Boolean expressions; . . . "

Converting logic circuits to Boolean expressions is standard practice, originating with Shannon's work in 1938. The patent in effect admits that this portion of the system is not novel, by describing one of the prior approaches as:

> "(ii) A method in which Boolean expressions for output parameters are produced from upper and lower logic, . . . " [Patent, Column 1]

Note that the terms "upper" and "lower" here mean abstract and detailed representations of the circuit, respectively.

Part 3. BDD generation

> ". . . a binary decision diagram (BDD) production device applying Shannon's formula to said produced Boolean expressions under a same order of variables to be extracted, and producing BDDs; . . . "

The method used to convert Boolean expressions to BDDs [Patent, Fig. 10A] is very similar to that described in [Akers78]. It involves repeatedly selecting a variable and considering the effects of substituting both 0 and 1 for this variable. Akers describes this as a

> ". . . 'top-down' procedure . . . to derive the diagram by repeated applications of the classical Shannon expansion formula . . . " [Akers78, p. 510]."

The only special feature in the Hitachi approach is that they select the variables in the same order when generating the BDDs for the two circuits to be compared. Akers alluded to this idea in his paper, stating that:

> "In the procedures of Section II, for example, we blindly processed the input variables in alphabetical order. Clearly other orders and techniques would lead to different and often simpler diagrams." [Akers78, p. 515]

In this statement, Akers expresses this restriction as a weakness in his examples. Considering only alphabetic ordering is indeed an undesirable limitation, but restricting to the same ordering for both BDDs greatly simplifies the comparison task.

Part 4. BDD simplification

> ". . . a BDD simplification device simplifying said BDDs; . . . "

They apply a series of node elimination rules [Patent, Fig. 3A], working from the bottom to the top of each BDD. The term "simplify" is used in a relative sense here. They do indeed eliminate nodes from the BDD, but not to the extent that they reduce the BDD to a minimal form. This will be discussed in greater detail when comparing to the uncited work.

Part 5. Equivalence test

> "... a comparator comparing said simplified BDDs and producing a signal in response to said comparing, said signal indicating said equivalence between said two logic circuits; ..."

This test involves two parts. First, the two BDDs are combined and reduced by a series of transformations applied from the bottom to the top of the BDDs. Each transformation identifies a class of isomorphic nodes and replaces them by a single element from the class. The actual test involves simply determining if the roots for the two BDDs reduce to the same node.

To aid in the discussion, we will split this stage into two parts: 5a being the reduction and 5b being the equality test.

Stage 5a, as described in the patent, is inadequate to guarantee that equivalent BDDs will reduce to having equal root nodes. In particular, the reduction process should also incorporate one of the transformations from stage 4. This error is not a fundamental flaw. In correcting this error, however, one would most likely incorporate the processing of steps 4 and 5a into a single simplification and reduction stage.

Part 6. Control

> "... a controller controlling a start and stop of said Boolean expression extractor, said BDD production device, said BDD simplification device and said comparator."

This is simply language to complete the description in terms of an apparatus.

Discussion

Figure 2 illustrates how the various BDD-based comparison techniques relate to one another. The first column illustrates the Hitachi system, while the second illustrates the approach taken by Akers, combining the BDD generation method of [Akers78] with the comparison method of [Akers80]. In this figure we omit the stages corresponding to parts 1–2 and 6 of the Hitachi system, as they simply reflect standard practice common to most logic comparison systems.

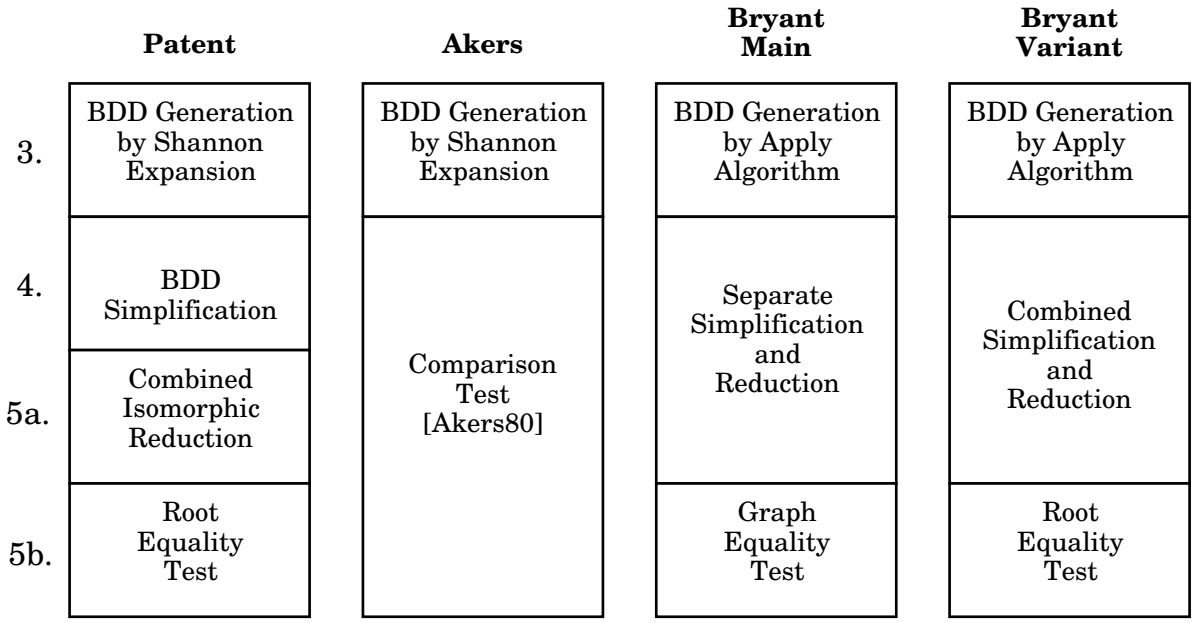

Figure 2: Stages in Different BDD-based Logic Comparison Techniques

The BDD generation phase (part 3) is essentially the same in both cases. However, the Hitachi approach for comparing two BDDs, comprising parts 4, 5a, and 5b of the system, differ significantly from the comparison method of [Akers80]. By exploiting the variable ordering restriction imposed during the BDD generation phase, they obtain a significantly more efficient approach.

5. Comparison to Uncited Work

We will use [Bryant86] as our basis for comparing the Hitachi approach to prior work not cited in the patent. Reference [Bryant85] is an earlier, somewhat abbreviated version of this paper. Reference [Madre88] describes the implementation of a logic comparison system at Bull in France using an extension of the algorithms.

Textual Analysis

A cursory inspection of [Bryant86] immediately indicates a close relation to the logic comparison technique described in the patent. In the abstract, it states:

> "Functions are represented by directed, acyclic graphs in a manner similar to Lee[1] and Akers[2], but with further restrictions on the ordering of decision variables in the graph." [Bryant86, p. 677]

As keywords it lists index terms: "binary decision diagrams" and "logic design verification." In the introduction section of the paper it makes the relation to Akers' work more explicit, stating:

> "Our representation resembles the binary decision diagram notation introduced by Lee[1] and further popularized by Akers[2]. However, we place further restrictions on the ordering of decision variables in the vertices. These restrictions enable the development of algorithms for manipulating the representations in a more efficient manner." [Bryant86, p. 677].

In fact, the restriction mentioned above is exactly that exploited by the Hitachi technique: that the same ordering be used for all of the BDDs.

A key result proved mathematically in [Bryant86] is that for a given ordering of variables, a BDD can be reduced to a unique form. This property is described in the introduction as:

> "... our representation in terms of reduced graphs is a canonical form, i.e., every function has a unique representation. Hence, testing for equivalence simply involves testing whether two graphs match exactly ..." [Bryant86, p. 678]

It is precisely this property that the Hitachi technique exploits.

Overall Approach to Logic Comparison

[Bryant86] places the task of comparing two logic circuits in a more general framework of creating

> "... representations of functions and determining various properties about them." [Bryant86, p. 681]

The particular procedure followed for logic comparison is illustrated schematically in the third column of 2. As with the Hitachi approach, BDD representations are generated from the logic circuit representations, and these BDDs are simplified to a reduced form and compared for equivalence. This process is described as follows:

> "... suppose we wish to construct the representation of the function computed by a combinational logic gate network. ... we proceed through the network, constructing [the representation of] the function computed at the output of each logic gate ... A similar procedure is followed to construct the representation of the function denoted by some Boolean expression. At this

> point we can test various properties of the function, such as ... whether it equals the function denoted by some other expression." [Bryant86, pp. 681–682]

Note that in the above text, constructing the "representation of the function" means generating a BDD representation.

BDD Generation

The technique for generating a BDD from a logic circuit or Boolean expression differs significantly in [Bryant86] from that of [Akers78] or [Patent]. The technique described actually improves on Akers' approach significantly. There are many cases where the complexity of Akers' approach will blow up exponentially, since the number of cases to be considered can double with each variable selected. This fact is even pointed out in the patent description of the BDD generation stage:

> "Substitute '0' and '1' to extracted variables, calculate Boolean expression, and store the resultant Boolean expression to temporary area *(the calculated expressions are double)* [Italics added]" [Patent, Fig. 10A, Step 302]

BDDs are generated in [Bryant86] by symbolically evaluating the network or expression using a new algorithm called "Apply." As stated in the paper:

> "The procedure *Apply* provides the basic method for creating the representation of a function according to the operators in a Boolean expression or a logic gate network." [Bryant86, p. 683]

Although this approach can also encounter exponential blow-up, this happens much less often than with Akers' approach. Variants of the Apply algorithm are now used in virtually all BDD-based circuit analysis programs.

Although the Apply algorithm differs significantly from that described in the patent, Claim 1 covers the general approach of "... a binary decision diagram production (BDD) device applying Shannon's formula to said produced Boolean expressions under a same order of variables to be extracted, and producing BDDs ...." The generation of BDDs using the Apply algorithm can be argued to fall within the scope of this claim. In particular, the operation of the algorithm is described as:

> "The control structure of the algorithm is based on the following recursion, derived from the Shannon expansion ..." [Bryant86, p. 684]

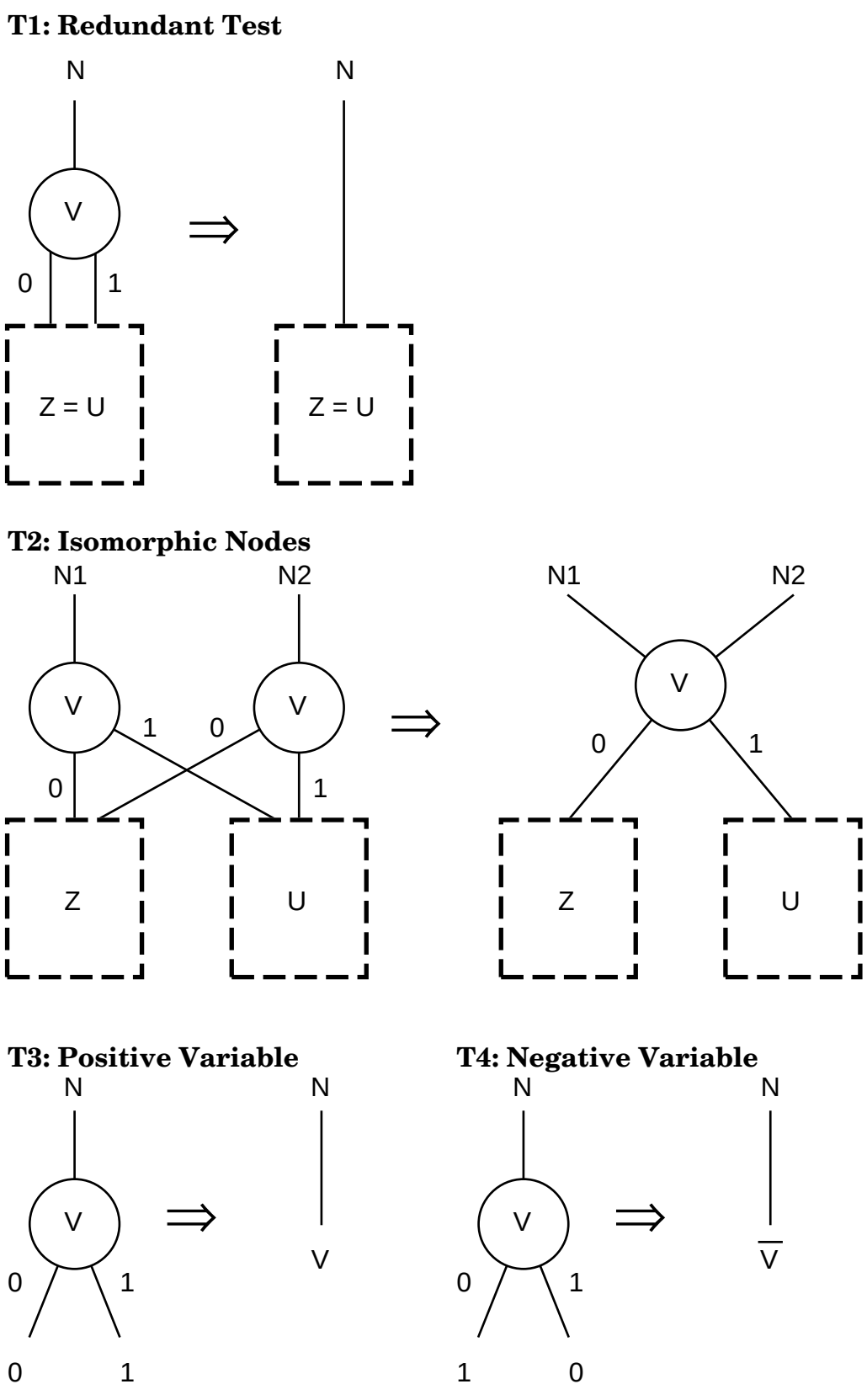

Figure 3: Transformations used in BDD simplification and reduction

Transformation Rules

Both the approach of [Patent] and of [Bryant86] determine whether two BDDs represent the same function by reducing them via a set of transformation rules into a form where the two BDDs, if equivalent, should become identical. This process is split over parts 4 and 5a of the system described in Claim 1 of the patent.

Figure 3 illustrates the transformation rules used in the Hitachi approach. In these figures, the rules show how a BDD containing one or more nodes labeled by a variable V can be simplified by eliminating the node. In these figures, the rectangular regions enclosed in dashed boxes represent other regions of the BDD. Only the first two transformation rules of Figure 3 are used in [Bryant86]. These rules can be explained as follows:

T1: Redundant test elimination Suppose the two branches (labeled as '0' and '1') leading out from a node go to the same destination. If, during an evaluation, we encounter such a node, we would take the same path regardless of the the value of V. Therefore the node can be eliminated and any incoming branch (labeled N) can instead be directed to the destination.

T2: Isomorphic node elimination If two or more nodes (labeled by branches N1 and N2) are *structurally identical*, i.e., they have the same variable and branch destinations, then all but one can be eliminated, and all branches leading to these nodes can be directed to the remaining one.

T3 and T4: Terminal variable recognition These rules apply to the variant of BDDs allowing the bottom-most branches of the BDD to point to variables or their complements. In the case where a node has destinations 0 and 1, this node can be replaced by a variable (positive case), or its complement (negative case).

Simplification and reduction in [Patent]

Rules T1, T3, and T4 are applied as part of the "BDD simplification" (part 4) [Patent, Fig. 3A]. These transformations are applied individually to the two BDDs to be compared. Figure 4 shows an example of two BDDs N1 and N2 (top), and the result of applying the simplification rules (middle).

Rule T2 is applied in part 5a prior to the actual equality test [Patent, Fig. 12, steps 502–504]. The transformation is applied to the two BDDs simultaneously, motivating the terminology "Combined Isomorphic Reduction" in the first column of Figure 2. That is, the structurally identical nodes from among the two BDDs are merged into a single node. The node with minimum address is chosen as the single representative. Combining and reducing the BDDs of Figure 4 yields the structure shown at the bottom. The three nodes labeled by variable B have been merged into a single node.

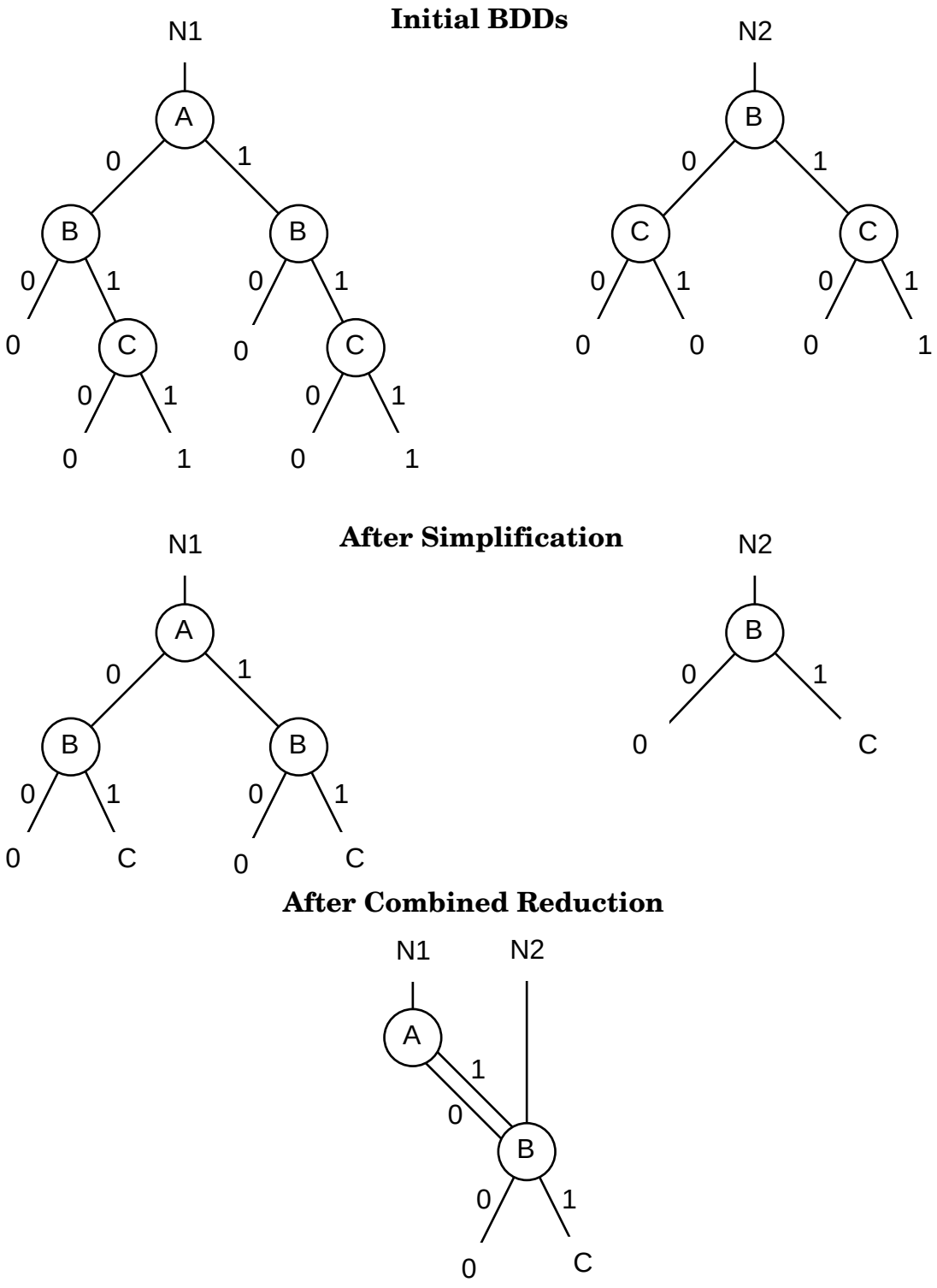

Figure 4: Simplification and reduction steps of patent illustrating error

Herein lies the error of the Hitachi approach. They should also apply rule T1 in part 5a when they are reducing the two BDDs. The transformations in this part can cause the two branch destinations leading from a node to be merged, and hence this node should be eliminated. This possibility actually occurs to the two branches leading from the node labeled by variable A in the example of Figure 4. Applying rule T1 to this graph would cause the node to be eliminated and branch N2 to be directed to the remaining node.

By failing to eliminate such nodes in part 5a, they can cause the verification process to fail, falsely declaring two equivalent BDDs to represent different functions. This error could be corrected by changing updating process described in the patent [Patent, Column 9]. More logically, they could incorporate all four transformations into a single process of eliminating and combining the nodes of the two BDDs to be compared.

Simplification and reduction in [Bryant86]

As mentioned earlier, [Bryant86] proves that, for a given variable ordering, any BDD can be reduced to a unique minimal form. The implementation of the reduction process is described as follows:

> "The reduction algorithm transforms an arbitrary function graph into a reduced graph denoting the same function." [Bryant86, p. 682].

Rather than eliminating nodes from the original BDD, the process proceeds in two parts. In the first part, the nodes are labeled as follows:

> "Proceeding from the terminal vertices up to the root, a unique integer identifier is assigned to each unique subgraph root." [Bryant86, p. 682]

That is, two nodes will be given the same label if they represent equivalent functions. After the labeling, the reduced BDD is generated:

> "Given this labeling, the algorithm constructs a graph with one vertex for each unique label." [Bryant86, p. 682]

Transformation rules T1 and T2 are embodied in the labeling process. This process is described in the paper for assigning an identifier $id(v)$ to a node v, where the node variable is identified by an "index" i. It labels a redundant test node with the same label as its branch destinations:

> "... if $id(low(v)) = id(high(v))$, then vertex v is redundant and we should set $id(v) = id(low(v))$." [Bryant86, p. 682]

It labels all isomorphic nodes by the same label:

> "...if there is some labeled vertex u with $index(u) = i$ having $id(low(v)) = id(low(u))$ and $id(high(v)) = id(high(u))$, then the reduced subgraphs rooted by these two vertices will be isomorphic, and we should set $id(v) = id(u)$." [Bryant86, p. 682]

The variant of BDDs used in [Bryant86] only permits constants 0 and 1 as the bottom-most branch destinations, and hence rules T3 and T4 are not applicable.

In the main presentation of [Bryant86], this process of transforming a BDD into its reduced form is carried out separately for the two BDDs to be compared. This motivates the terminology "Separate Simplification and Reduction" in the third column of Figure 2.

Equality Test

Since the isomorphic reduction in the Hitachi approach (part 5a) is performed on the combination of the two BDDs being compared, the root nodes should merge into a single node, assuming the BDDs are indeed equivalent. Hence, part 5b simply involves testing whether the two roots have become identical. This motivates the terminology "Root Equality Test" in the first column of Figure 2. Of course, our example of Figure 4 illustrates that this test would suffice only if rule T1 were also applied in part 5a.

Since the reduction in the main presentation of [Bryant86] is performed on the BDDs separately, the two reduced BDDs must still be compared to one another to test for equivalence. This motivates the terminology "Graph Equality Test" in the third column of Figure 2. This test is quite efficient, since the two graphs should match exactly.

Discussion

As we have presented, the approach for simplifying and comparing BDDs described in [Patent] and [Bryant86] differ in the following respects:

- The process of reducing a BDD to its unique form is split across parts 4 and 5a in [Patent], whereas it is performed by a single reduction algorithm in [Bryant86].
- The BDD of [Patent] is slightly more general than that of [Bryant86], necessitating additional transformation rules T3 and T4.
- The two BDDs to be compared are reduced simultaneously in [Patent], but separately in [Bryant86].

- The final equality test requires only comparing the roots in [Patent], but involves the comparing the two BDDs in [Bryant86].

Claim 1 of the patent is broader than the particular approach described in the patent. It is clear that the reduction algorithm of [Bryant86] realizes a "BDD simplification device," while the graph equality test realizes a "comparator comparing said simplified BDDs . . . indicating said equivalence . . . ." The work described in [Bryant86] fully realizes the system described in this claim.

Variant Approach

Although the main presentation of [Bryant86] reduces each graph separately, the idea of a combined reduction is mentioned in the concluding remarks:

> "Alternatively, we could represent a set of functions by a single graph with multiple roots (one for each function.) Our reduction algorithm could be applied to such graphs to eliminate any duplicate subgraphs and to guarantee that the subgraph consisting of a root and all of its descendants is a canonical representation of the corresponding function." [Bryant86, p. 689].

This approach has, in fact, become standard in most implementations of BDDs. A system for logic comparison incorporating this variant is diagrammed in the fourth column of Figure 2. As can be seen, the distinction between this variant and the system described in the patent becomes negligible.

6. Other Claims in Patent

The patent contains a total of ten claims, of which we have discussed the first one in depth. Claims 2–5 place further restrictions on the system of Claim 1. Such restricted claims are added to a patent in the hope of preserving some of the patent in the event that a broader claim is found invalid.

Claim 2 restricts the comparison part of the system such that:

> ". . . said comparator integrates and compares said two BDDs simplified by said BDD simplification device from branches."

That is, they make explicit the combined isomorphic reduction of part 5a. The variant method of [Bryant86] fulfills this claim.

Claim 3 restricts the comparison task to one:

> "... wherein said first and second logics are an upper level logic and a lower level logic in a hierarchy logic design."

That is, one circuit is in a more abstract form than the other. Comparing descriptions at different levels of abstraction is a common use of logic verification. For example, [Bryant86] presents results on the following experiment:

> "For this paper, we consider the problem of verifying that the implementation of a logic function (in terms of a combinational logic gate network) satisfies its specification (in terms of Boolean expressions.)" [Bryant86, p. 687].

Claim 4 restricts the system to one

> "... wherein said system is included within a computer."

This restriction is included due to controversies over the patentability of algorithms. In the past, the Patent Office has held that a process that could be construed as a series of "mental steps" could not be patented, but that the computer implementation of such a process could be. [OTA93, p. 46].

Claim 5 restricts the system as follows:

> "... further comprising a storage device storing Boolean expressions of parameter forms corresponding to logic circuit elements of said two logic circuits, wherein said Boolean expression extractor converts said stored first and second logics to Boolean expressions ..."

In more common parlance, they mean that they maintain a library of Boolean expressions in conjunction with their library of standard circuit modules. The overall circuit behavior is computed by composing the expressions according to the module interconnections. Again, one can argue that this is standard practice for most logic comparison systems. For example, [Bryant86] describes a similar process in the experimental results section:

> "For our experiments, we derived the functions for the two chips from their gate-level descriptions and then composed these functions to form the different ALU's according to the chip-level interconnections in the circuit manual." [Bryant86, p. 688]

Claim 6 describes the comparison as a "method," i.e., a process, rather than a "system," i.e., an apparatus. The standards for patentability are somewhat different for these two forms of inventions [OTA93, p. 132]. The description closely matches

that of Claim 1, but as a sequence of steps rather than as a set of components. Furthermore, Claim 6 incorporates the notion of a parameterized library analogous to that of Claim 5.

Claims 7–9 restrict the method of Claim 6 in a manner analogous to Claims 2–4, respectively.

Finally, Claim 10 reiterates the method of Claim 6, giving a slightly less detailed description. It makes explicit the fact that the approach applies to circuits "... designed by a plurality of methods or by a plurality of designers ...." This generality is also observed in the discussion of experimental results in [Bryant86]:

> "We have implemented the algorithms described in this paper and have applied them to problems in logic design verification, test pattern generation, and combinatorics. On the whole, our experience has been quite favorable. By analyzing the problem domain, we can generally develop strategies for choosing a good ordering of the inputs. ... Functions rarely require graphs of size exponential in the number of inputs, as long as a reasonable ordering of the inputs has been chosen. In addition, the algorithms are quite fast, remaining practical for graphs with as many as 20 000 vertices." [Bryant86, p. 687]

None of these additional claims affects the central issue of whether the approach described in the patent improves on the one described in [Bryant86].

7. How Did This Happen?

One can only conjecture why the patentees as well as the patent examiner failed to cite the relevant publications described here. It seems quite clear, however, that the patentees developed their approach independently of the work described in [Bryant86]. This conclusion is based on the following observations:

- They use the older, less efficient BDD extraction technique of [Akers78].
- They use the variant of BDDs described in [Akers78], rather than that of [Bryant86].
- They split the task of reducing the BDDs to canonical form over two parts of the system.
- Their approach contains a flaw not found in [Bryant86].

One concludes, therefore, that the Hitachi researchers were either unaware of or did not understand the presentation in [Bryant86]. Apparently, the patent examiner did not uncover any of these relevant papers during the examination process either.

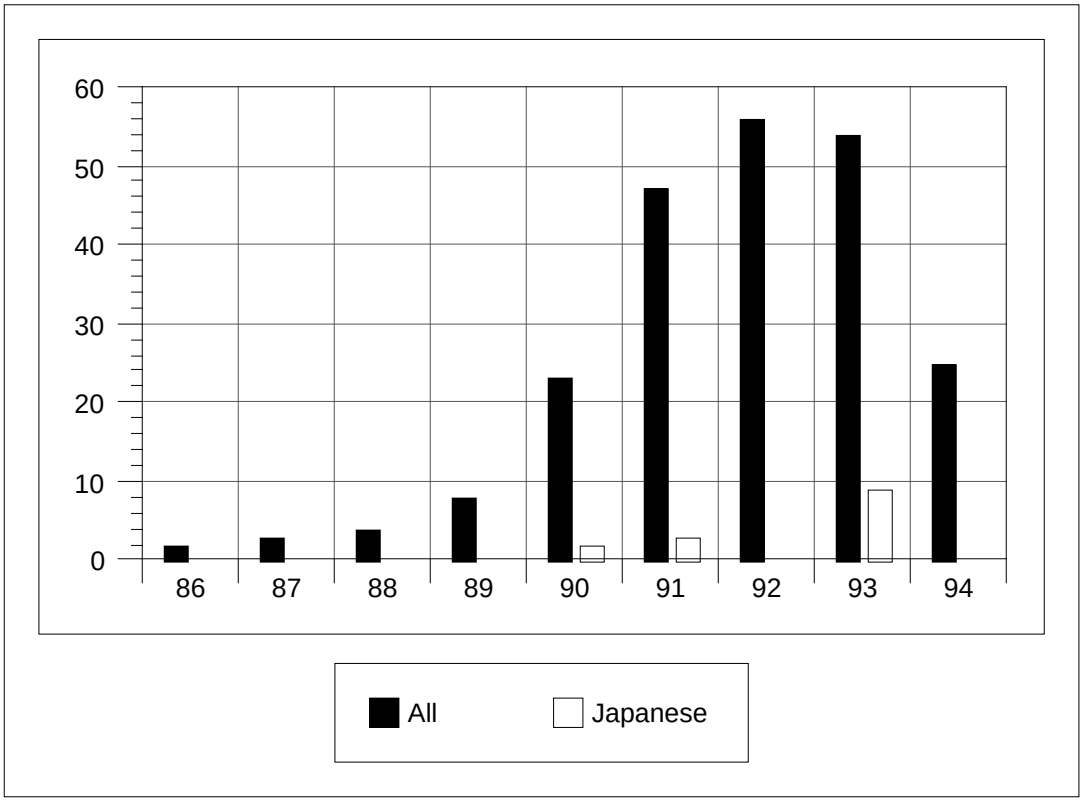

Figure 5: Keyword matches from INSPEC database

This failure to cite related technical literature provides an interesting case study in the challenges of disseminating research results and of technology transfer. How could people developing a program for analyzing digital circuits fail to uncover relevant papers presented at the largest annual conference focussing on electronic design automation [Bryant85], or published in a journal produced by the largest professional society for electrical and computer engineering [Bryant86]?

Database Analysis

To gain a more in-depth understanding of this issue, I have attempted to measure the dissemination of work on binary decision diagrams via several databases. The first is the INSPEC database, allowing keyword search for papers appearing in a wide range of engineering publications, including many conferences. This database consists primarily of publications from 1988 onward, although there are some earlier ones. Searching with a keyword query that would match both "binary decision diagram" and "binary decision diagrams" yields 222 citations (as of August 16, 1994). Figure 5 breaks these down according to the year of publication, showing both citations for all languages as well as those for articles in Japanese. As can be seen, activity in the field did not really take off until around 1990. Note that the apparent downturn starting in 1993 reflects the backlog for entering citations into the database,

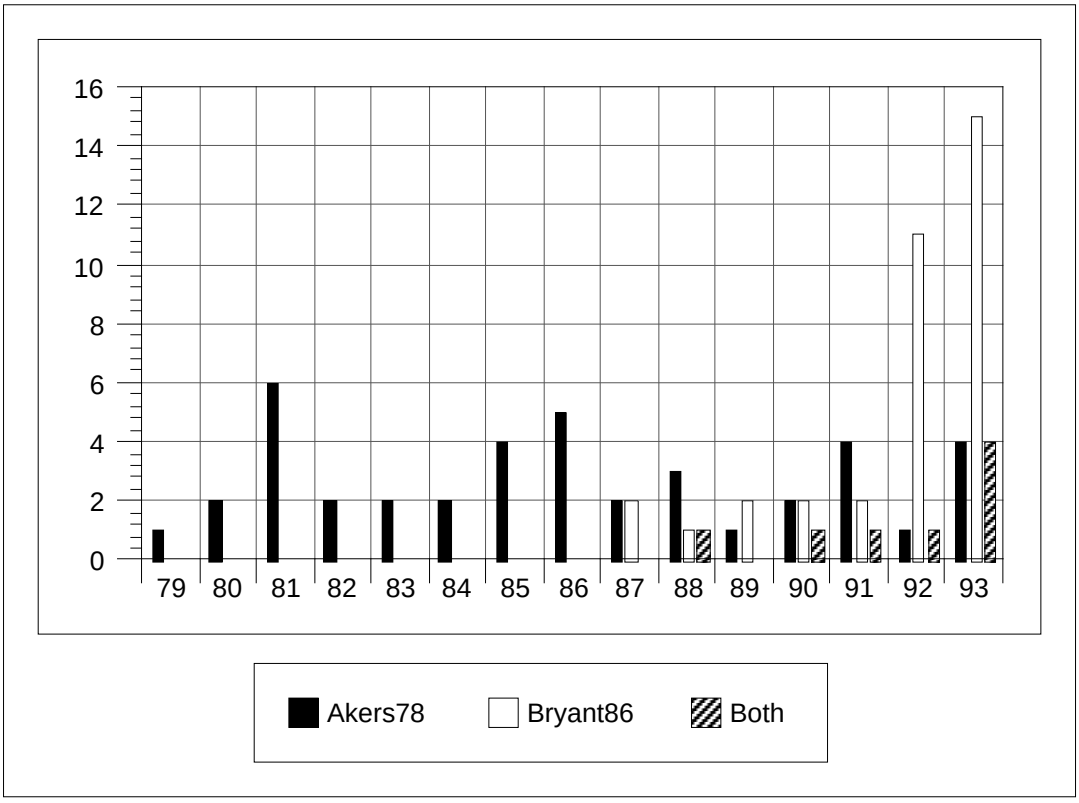

Figure 6: Citations in Science Citation Index

rather than decreased activity in the area.

This search also demonstrates some of the limitations of finding relevant literature by keyword search. Papers [Bryant85] and [Bryant86] are not found, because they are too early for the database. Paper [Madre88] is not found, because the paper refers to the graphs as "typed Shannon's canonical forms," rather than BDDs. Restricting the search to only consider papers up to August, 1988 (one year before the filing of the Japanese patent), one finds only nine total citations. Three of these cite and make use of the results of [Bryant86], but none discuss applications to logic comparison or verification.

The second database is the Science Citation Index (SCI), providing a cross-index to all papers citing a particular reference. This index only catalogs citations from journal publications. This is a significant limitation, as much of the research in the field is disseminated through conference publications. Figure 6 shows the citation history for publications [Akers78] and [Bryant86]. Observe that the citations to [Akers78] are steady, but at a relatively low level, averaging 2.7 per year. Citations to [Bryant86], on the other hand, occur at a low level up through 1991, and then climb sharply. In fact, most of the recent citations of [Akers78] also cite [Bryant86], as indicated by the series labeled "Both."

Journal publications have a significant time delay due to the reviewing process and

the publication backlog. Hence, one would expect the citations to start building at least 2 years after work is published. The delay of around 5 years cannot be completely explained on this basis.

Qualitative analysis

New approaches to solving problems generally take hold more slowly than extensions to existing approaches. The approach to logic verification described in [Bryant86] differs significantly from previous methods. As the citation history shows, the paper generated relatively little interest at first. In fact, many well-respected researchers did not fully understand the approach or its capabilities.

Interest in BDD-based approaches to verification began to grow only after the presentation of two papers at the International Conference in Computer-Aided Design in November, 1988 [Fujita88, Malik88]. These papers presented heuristic methods for solving the difficult problem of finding a good variable ordering for the BDDs and demonstrated the capability to deal with very large benchmark circuits. Most likely the Hitachi researchers developed their approach prior to this date.

Patent examination

One may question why the patent examiner was not familiar with the relevant literature on BDD-based logic comparison. By the time he would have seen the application, there were ample publications on the subject. One might expect an employee of the Patent Office examining patents on computer-aided design would consult the publications of the leading conferences and journals in the subject area. Furthermore, one might expect examiners to use databases such as INSPEC and SCI to search for relevant literature.

Sadly, neither of these appear to be the case. Until this year, training in computer science was not considered a valid background for employment as a patent examiner [NYT94]. Thus, patent examiners tend to have a limited understanding of the basic concepts of data structures and algorithms. Furthermore, they look for relevant prior art mostly by searching their patent database [NYT94]. Since patenting of software has only recently become a standard practice, the patent literature provides a very incomplete picture of the prior art in computer-aided design.

8. Final Thoughts

Through my training as a computer science researcher, I was generally under the impression that work such as that reported in [Bryant86] was not patentable. Indeed the issue of whether an invention that contains a “mathematical algorithm” can be

patented remains problematic [PTO89]. As the title of the 1986 paper [Bryant86], I viewed its main contribution as a method to create and manipulate representations of abstract mathematical objects. Indeed, under current patent guidelines, a mathematical algorithm devoid of an application is not patentable. On the other hand, I also viewed the work as providing a means of solving real-world problems, such as logic comparison, and I even validated the ideas by implementing a prototype system. As evidenced by the granting of this patent, it would have been possible to obtain a patent within the context of such an application. At the time I did this research, the thought of applying for a patent hardly crossed my mind.

Beyond the issue of whether one *could* obtain a patent on an algorithm, there remains the question of whether one *should* do so. Among my colleagues in academia, I find widely varying opinions and even degrees of interest on this issue. Some argue that patents inhibit the free dissemination of research results and hence are contrary to the purpose of universities. Others argue that the royalty income derived from patents will promote further innovation and research. A third group claims that the potential economic benefit of a patent is almost always outweighed by the time, effort, and cost of obtaining it and of pursuing possible infringers. A final group expresses general disinterest in the subject, being more concerned with such measures of success as obtaining tenure and peer recognition. It is clear that the prevailing attitude is shifting toward obtaining patents and other forms of intellectual property protection for research performed in universities.

References

[Akers78] S. B. Akers, "Binary Decision Diagrams," *IEEE Transactions on Computers*, Vol. C-27, No. 6 (June, 1978), pp. 509–516.

[Akers80] S. B. Akers, "A Procedure for Functional Design Verification," *10th International Symposium on Fault-Tolerant Computing*, 1980.

[Bryant85] R. E. Bryant, "Symbolic Manipulation of Boolean Functions Using a Graphical Representation", *22nd ACM/IEEE Design Automation Conference*, June, 1985, pp. 688–694.

[Bryant86] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, Vol. C-35, No. 8 (August, 1986), pp. 677–691.

[Fujita88] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluations and Improvements of a Boolean Comparison Program Based on Binary Decision Diagrams," *International Conference on Computer-Aided Design*, November, 1988, pp. 2–5.

[Madre88] J. C. Madre and J. P. Billon, "Proving Circuit Correctness Using Formal Comparison between Expected and Extracted Behaviour," *25th ACM/IEEE Design Automation Conference*, June, 1988, pp. 205–210.

[Malik88] S. Malik, A. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment," *International Conference on Computer-Aided Design*, November, 1988, pp. 6–9.

[NYT94] S. Chartrand, "Patents; Ideas, Advice and Criticism Spring Forth on How, and Whether, to Grant Patents Involving Software," *New York Times*, Feb. 14, 1994, Sec. D, p. 2.

[OTA93] *Computer Software: Copyrights, Patents, and the Challenge of Technological Change*, Office of Technology Assessment, United States Congress, 1993.

[Patent] U.S. Patent #5,243,538, "Comparison and Verification System for Logic Circuits and Method Thereof," Issued Sept. 7, 1993.

[PTO89] "Mathematical Algorithms," *Official Gazette of the United States Patent and Trademark Office*, Vol. 1106, No. 1 (Sept. 5, 1989), pp. 5–12.