Volume 2 ▪ Issue 2 ▪ August 2011

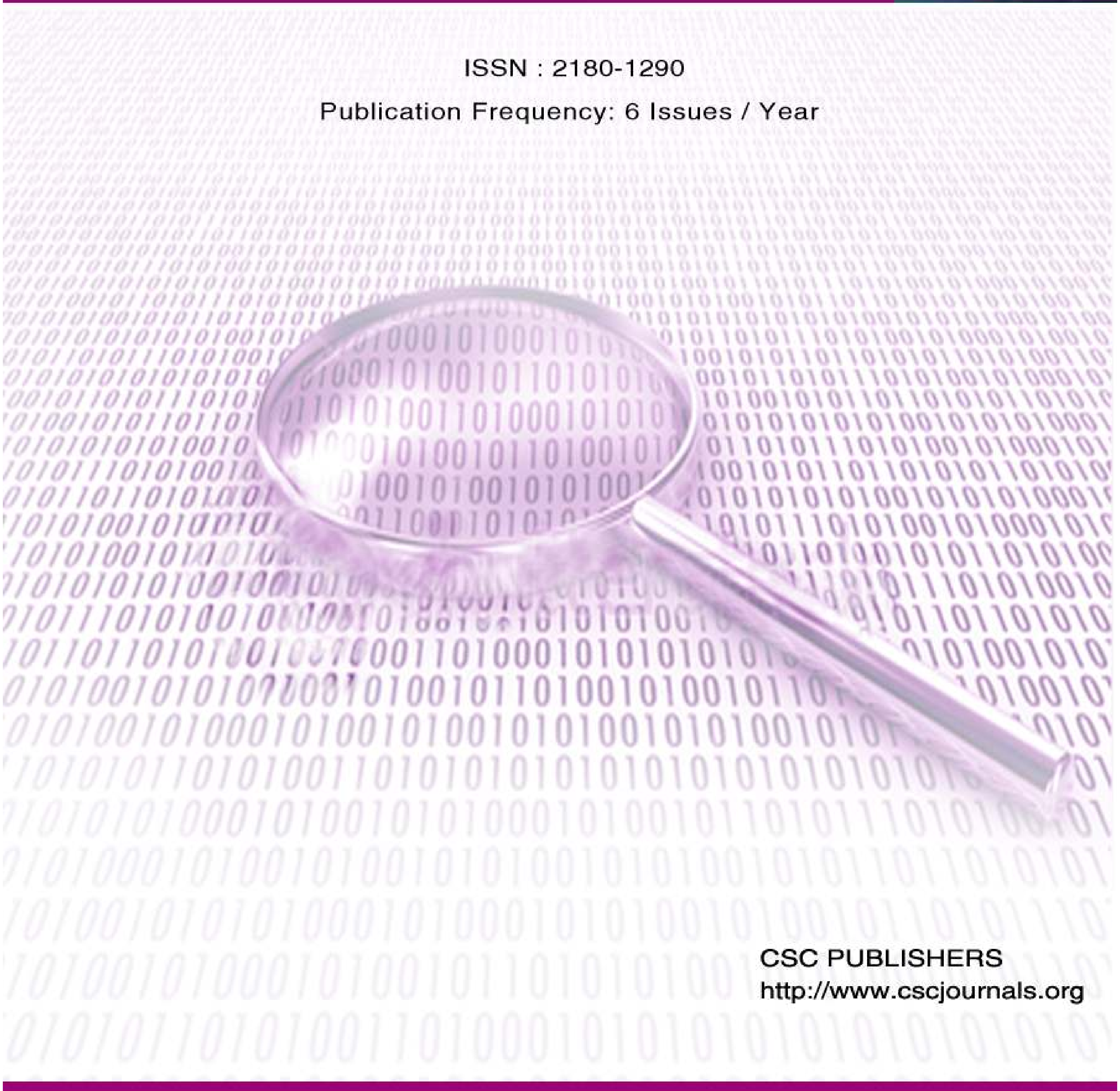
Editor-in-Chief
Professor. Santosh Kumar Nanda

INTERNATIONAL JOURNAL OF

LOGIC AND COMPUTATION (IJLP)

ISSN : 2180-1290

Publication Frequency: 6 Issues / Year



CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF LOGIC AND COMPUTATION (IJLP)

VOLUME 2, ISSUE 2, 2011

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 2180-1290

International Journal of Logic and Computation (IJLP) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJLP Journal is a part of CSC Publishers
Computer Science Journals
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF LOGIC AND COMPUTATION (IJLP)

Book: Volume 2, Issue 2, August 2011
Publishing Date: August 2011
ISSN (Online): 2180-1290

IJLP Journal is a part of CSC Publishers
http://www.cscjournals.org

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2011

EDITORIAL PREFACE

It is a great privilege for me as Editor in Chief of International Journal of Logic and Computation (IJLP) to present our readers the current issue of Journal which wraps up its first year and first issue of successful publication. This journal has focused on publishing research that provides information for practitioners, researchers and academicians with a teaching or research interest in engineering and science discipline. The first issue of IJLP is organized to presents articles in a particular area of computer logic and computation to attract readers who are interested in reading papers related to that special field. The first issue of IJLP provides a better chance to fulfill the anticipation of a broader community of our audiences.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2, 2011, IJLP appears in more focused issues. Besides normal publications, IJLP intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

As EIC of IJLP, I want to encourage contributors to IJLP to submit not only manuscripts addressing basic and applied research articles but also reviewed articles, practitioner oriented papers and other exploratory research projects addressing contemporary issues in such areas as Computational Logic, Knowledge based systems, Application of Logic in Hardware and VLSI, Soft Computing Techniques, Type theory, Natural Language etc. The review process will remain the same for these articles as mentioned in the official website of IJLP.

IJLP editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build international reputation of IJLP, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc, Scribd, CiteSeerX and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJLP. I would like to remind you that the success of the journal depends directly on the number of quality articles submitted for review. Accordingly, I would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits that IJLP editors provide to the prospective authors is the mentoring nature of the review process. IJLP provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members
International Journal of Logic and Computation (IJLP)

TABLE OF CONTENTS

Volume 2, Issue 2, August 2011

Time Table Scheduling Problem Using Fuzzy Algorithmic Approach

Poornima B *poornima_teju@rediffmail.com*
Department of Computer Science and Engineering
B I E T
Davangere – 577 004, Karnataka, India.

Dr. V. Ramaswamy *researchwork04@yahoo.com*
Department of Computer Science and Engineering
S B M J C E
Bangalore, Karnataka, India

Abstract

In this paper we develop an algorithm to generate a course Time table using fuzzy algorithmic approach satisfying certain constraints. With an example we show that how these constraints are satisfied.

Keywords: Time Tabling Problem, Fuzzy Algorithmic Approach, Fuzzy Membership Values.

1. INTRODUCTION

Time tabling problem exists since time immemorial. No academic Institute can function without a proper time table. Despite the problem being quite old, it is also a very challenging problem. Time tabling of an Institute depends to a large extent on the Institute and the various resources available in the Institute. Hence any attempt at developing a generic solution to the time tabling problem will come to naught. At the same time, it is also true that several papers [1] which deal with finding solutions to time tabling problem using various algorithms are appearing on a continuous basis.

Time tabling problem involves factors [2] such as teachers, classes and courses. Various resources are rooms, time slots etc. Time tabling problem is concerned with maximum utilization of the available resources subject to a set of constraints [3] and can be classified into three main classes namely school time tabling, course time tabling and examination time tabling.

In this paper, we have attempted to give solution to course time tabling problem based on algorithmic approach in which priorities of teachers for the various classes taught by them are considered using fuzzy membership values. Fuzzy set theory introduced by Prof. Lotfi A. Zadeh of Berkeley in 1965 as a means to model vagueness and ambiguity in complex systems is gaining more [4] and more popularity with every passing day. Today, practically there is no field in which fuzzy techniques do not play any role.

Ordinary sets (referred to as crisp sets) are applicable only in cases where there is absolute certainty. But our life is not full of certainties. In our every day life, we use vague terms such as old, young, short and tall more often than precise terms such as 75 years and 5 months, 22 years and 7 ½ months, 4′ 7″, 6′ 3″ etc. These terms are referred to as fuzzy terms.

Let X be a universal set. A fuzzy set A on X is characterized by a membership function μ_A which is a function from X to [0, 1]. For an element $x \in X$, if $\mu_A(x) = 0$, then $x \notin A$. If $\mu_A(x) = 1$, then $x \in A$. As $\mu_A(x)$ moves closer to 0, the chances that x belongs to A are lesser and lesser. As $\mu_A(x)$ moves away from 0 and closer to 1, the chances that x belongs to A are more and more. μ_A need not take only real values between 0 and 1. μ_A can assume fuzzy values such as low, medium, high etc. In this paper, we generate time table by assigning membership grades for the priorities given by a teacher for different time slots.

We are assuming the following constraints. Nevertheless, the solutions we propose can be applied to general time tabling problem.

1. We assume that the number of semesters is three.
2. Six subjects are to be taught by different teachers for each semester.
3. Every teacher will be handling two subjects (of course, for different semester).
4. Each subject should be taught for four hours in a week.
5. Classes are held for seven hours from Monday to Friday and for four hours on Saturdays.

We note that the department requires totally nine teachers for teaching all the subjects and the department has to allocate time slots for 72 hours in a week.

2. FUZZY ALGORITHMIC APPROACH

In this approach, time table is generated by assigning membership grades for the priorities given by a teacher for different time slots. In our time tabling problem, we assume there are nine teachers T_1, $T_2 \ldots T_9$ who teach for three semesters S_1, S_2, S_3. Each teacher T_i teaches a subject for semester S_j for four periods in a week.

Suppose $(T_i, S_j, t_k) = h$. This means the degree to which teacher T_i takes class for semester S_j at time t_k is high. We can always find an r such that $7(r - 1) < k \leq 7r$. Then for any $p \neq j$ (in this example, there can be only one such p because any teacher teaches only two subjects), following conditions should hold good.

(i) $(T_i, S_j, t_k) = 0$ $(i \neq i)$ (Note that there are 5 more such i's corresponding to other five teachers for the class S_j)

(ii) $(T_i, S_p, t_k) = 0$. (Because it corresponds to the same slot for another semester for the same teacher).

(iii) $(T_i, S_p, t_{k-1}) = l$ ($1 < k$ and k not congruent to 1 (mod 7)).

This means $k \neq 8, 15, 22, 29, 36$. Note that if $k = 8$, then $(T_i, S_j, t_8) = h$ does not mean $(T_i, S_p, t_7) = l$. This applies to k = 15, 22, 29 and 36. A teacher can take class on the first hour of a day and the last hour of the previous day.

(iv) $(T_i, S_p, t_{k+1}) = l$ ($k \leq 38$ and k not congruent to 0 (mod 7). This means k is not a multiple of 7 ie. $7(r - 1) < k < 7r$).

(v) $(T_i, S_p, t_{k-2}) = m$ ($2 < k$, k not congruent to 2 (mod 7) and k not congruent to1(mod 7) ie. $k \neq 8, 9, 15, 16, 22, 23, 29, 30, 36, 37$).

(vi) $(T_i, S_p, t_{k+2}) = m$ ($k \leq 37$ and k not congruent to 0 (mod 7) this means $7(r - 1) < k < 7r$. Also $(k + 1)$ not congruent to 0 (mod 7). This means $k \neq 6, 13, 20, 27, 34$).

(vii) $(T_i, S_p, t_{k-q}) = h$ ($k > 3$, $3 \leq q < k - 7(r - 1)$).

(viii) $(T_i, S_p, t_{k+q}) = h$ ($k \leq 36$, $k \leq 7r - 3$, $3 \leq q$).

First fix the time table for (T_1, S_1) by placing 1s in convenient locations. There should be four 1s in each row. If there is a 1 for a teacher, for a semester in one day, all other entries for that teacher for that semester for that day should be made 0s. In other words, a teacher teaches a subject for a given semester in a given day for maximum one hour. Some hs should ultimately be converted to 1s satisfying the above conditions. Then the final time table would have been generated.

We assume that every class lasts for one hour. Also on week days, classes start at 8 am and end at 12 noon for the morning session (4 hours). Again, classes start at 2 pm and end at 5 pm for the afternoon session (3 hours). On Saturdays, classes are held only from 8 am to 12 noon. Thus there are 39 hours t_1, t_2,.......t_{39} where t_1 corresponds to 8 am to 9 am of Monday, t_7 corresponds to 4 pm to 5 pm of Monday, t_8 corresponds to 8 am to 9 am of Tuesday and finally t_{39} corresponds to 11 am to 12 noon of Saturday.

We now construct a matrix of size 18×39 as follows.
The first row of the matrix corresponds to (T_1, S_1). Last row corresponds to (T_9, S_2). The columns of the matrix correspond to t_1, t_2, t_3........t_{39}. Note that for each teacher T_i ($1 \leq i \leq 9$), there are two rows which correspond to the two semesters handled by T_i. We thus have an 18×39 matrix whose entries can be l which stands for low, m which stands for medium and h which stands for high. For example, if the degree to which teacher T_1 teaches semester S_1 at time t_1 is high, then we represent it by writing (T_1, S_1, t_1) = h.

2.1 Example 2.1
There are nine teachers T_1, T_2,.....T_9 and three semesters S_1, S_2 and S_3. Assume that teachers T_1, T_3, T_5 teach for semesters S_1 and S_3, teachers T_2, T_6, T_8 teach for semesters S_2 and S_3, teachers T_4, T_7, T_9 teach for semesters S_1 and S_2.

Suppose teacher T_1 teaches semester S_1 to a high degree when k takes the values 2, 10, 20, 26, 29, 39. Teacher T_2 teaches semester S_2 to a high degree when k takes the values 1, 10, 20, 26, 32, 37. T_3 teaches semester S_1 to a high degree when k takes the values 3, 11, 19, 28, 30, 36. T_4 teaches semester S_1 to a high degree when k takes the values 1, 9, 20, 25, 35, 37. T_5 teaches semester S_1 to a high degree when k takes the values 3, 13, 15, 23, 33, 39. Teacher T_6 teaches semester S_2 to a high degree when k takes the values 5, 8, 21, 27, 32, 38. Teacher T_7 teaches semester S_1 to a high degree when k takes the values 7, 11, 15, 27, 33, 37. Suppose teacher T_8 teaches semester S_2 to a high degree when k takes the values 2, 13, 21, 26, 29, 38. Teacher T_9 teaches semester S_1 to a high degree when k takes the values 1, 10, 21, 27, 33, 37. The entries of the matrix generated as follows.

For example, when T_1 teaches semester S_1 and k takes the value 2, enter (T_1, S_1, t_2) = h which means T_1 teaches class S_1 to a high degree. Then (T_3, S_1, t_2) = (T_4, S_1, t_2) = (T_5, S_1, t_2) = (T_7, S_1, t_2) = (T_9, S_1, t_2) = 0. Also (T_1, S_3, t_2) = 0. Thus conditions (i) and (ii) are satisfied.

The value of r depends on the value of k. If the value of k is a multiple of 7 i,e when k takes the value 7, 14, 21, 28 and 35, $r = \lfloor k / 8 \rfloor + 1$. Otherwise, the value of r is $\lfloor k / 7 \rfloor + 1$. Here the value of k is 2 and is not a multiple of 7. So $r = \lfloor 2 / 7 \rfloor + 1 = 1$. Then for S_3, other entries are computed as follows.

(T_1, S_3, t_1) = l because ($1 < 2$ and 2 not congruent to 1 (mod 7)) according condition (iii). (T_1, S_3, t_3) = l because 2 is not a multiple of 7. ie. $0 < 2 < 7$ satisfying the condition (iv). (T_1, S_3, t_4) = m because ($2 \leq 37$ and 2 not congruent to 0 (mod 7), satisfying the condition (v). (T_1, S_3, t_5) = h because ($2 \leq 36$, $2 \leq 4$) satisfying the condition (vi) and the value of q is 3. Correspondingly, (T_1, S_1, t_5) = 0, (T_1, S_1, t_6) = l (T_1, S_1, t_4) = l, (T_1, S_1, t_7) = m, (T_1, S_1, t_3) = m. All other values are made zéro i,e (T_1, S_1, t_1) = 0, (T_1, S_3, t_6) = 0, (T_1, S_3, t_7) = 0. Similarly generate entries for all the nine teachers for all the slots satisfying the conditions from (i) to (viii). For the above example, the matrix filled with ‘h’, ’m’ and ‘l’ is generated and is as shown in the table given in table below.

K varies	1 to 7	8 to 14	15 to 21	22 to 28	29 to 35	35 to 39
(T_1,S_1)	0hml0lm	00hml0l	ml0lmh0	l0lmh00	hml0lmh	0lmh
(T_1,S_3)	l0lmh00	ml0lmh0	00hml0l	mhml0mm	0lmhml0	hml0
(T_2,S_2)	hml0lmh	00hml0l	ml0lmh0	l0lmh00	000hml0	0h00
(T_2,S_3)	0lmhml0	ml0lmh0	00hml0l	mhml0lm	0ml0lmh	l0lm
(T_3,S_1)	00hml0l	m00hml0	l0lmh00	0ml0lmh	0hml0lm	hml0
(T_3,S_3)	ml0lmh0	0ml0lmh	0hml0lm	000hml0	l0lmhm0	0lmh
(T_4,S_1)	hml0lmh	0hml0lm	ml0lmh0	000hml0	0ml0lmh	0h00
(T_4,S_2)	0lmhml0	l0lmh00	00hml0l	mml0lmh	000hml0	l0lm
(T_5,S_1)	00hml0l	ml0lmh0	hml0lmh	0hml0lm	l0lmhm0	0lmh
(T_5,S_3)	ml0lmh0	00hmlml	0lmhml0	l0lmh00	0hml0lm	hml0
(T_6,S_2)	l0lmh00	hml0lmh	0ml0lmh	ml0lmh0	000hml0	00h0
(T_6,S_3)	0hml0mm	0lmhml0	000hml0	00hml0l	mml0lmh	ml0l
(T_7,S_1)	0ml0lmh	000hml0	hml0lmh	ml0lmh0	l0lmh00	0h00
(T_7,S_2)	000hml0	0ml0lmh	0lmhml0	00hml0l	mhml0lm	l0lm
(T_8,S_2)	0hml0lm	ml0lmh0	0ml0lmh	l0lmh00	hml0lmh	00h0
(T_8,S_3)	l0lmh00	00hml0l	m00hml0	0hml0mm	0lmhml0	ml0l
(T_9,S_1)	hml0lmh	00hml0l	mml0lmh	ml0lmh0	l0lmh00	0h00
(T_9,S_2)	0lmhml0	ml0lmh0	000hml0	00hml0l	mhml0lm	l0lm

TABLE 1. Matrix filled with 'h' 'm' and 'l'

Once the matrix is completed with all its entries, generate another matrix first by replacing all 'h' by '5' as shown in the table given below.

K varies	1 to 7	8 to 14	15 to 21	22 to 28	29 to 35	35 to39
(T_1, S_1)	0500000	0050000	0000050	0000500	5000005	0005
(T_1, S_3)	0000500	0000050	0050000	0500000	0005000	5000
(T_2, S_2)	5000005	0050000	0000050	0000500	0005000	0500
(T_2, S_3)	0005000	0000050	0050000	0500000	0000005	0000
(T_3, S_1)	0050000	0005000	0000500	0000005	0500000	5000
(T_3, S_3)	0000050	0000005	0500000	0005000	0000500	0005
(T_4, S_1)	5000005	0500000	0000050	0005000	0000005	0500
(T_4, S_2)	0005000	0000500	0050000	0000005	0005000	0000
(T_5, S_1)	0050000	0000050	5000005	0500000	0000500	0005
(T_5, S_3)	0000050	0050000	0005000	0000500	0500000	5000
(T_6, S_2)	0000500	5000005	0000005	0000050	0005000	0050
(T_6, S_3)	0500000	0005000	0005000	0050000	0000005	0000
(T_7, S_1)	0000005	0005000	5000005	0000050	0000500	0500
(T_7, S_2)	0005000	0000005	0005000	0050000	0500000	0000
(T_8, S_2)	0500000	0000050	0000005	0000500	5000005	0050
(T_8, S_3)	0000500	0050000	0005000	0500000	0005000	0000
(T_9, S_1)	5000005	0050000	0000005	0000050	0000500	0500
(T_9, S_2)	0005000	0000050	0005000	0050000	0500000	0000

TABLE 2. 'h' replaced by '5'

Next step is to generate a matrix in which each row will have four selected periods. The entries which satisfy the following conditions are changed from '5' to '2' to indicate that they are selected periods.

(1) Maximum one class in a day for the same subject.
(2) Maximum one 8 - 9 class in a week for each subject.
(3) Ultimately, there should be only four selected periods in any row.

As an entry is changed from ‘5’ to ‘2’ in a row, corresponding time slot for the same teacher for other semester (to which he teaches) is made zero indicating that he is not available for that semester during that time slot. Corresponding time slot for each of the other teachers who are teaching for the same semester is made zero indicating that the particular class is not free for any of them.For example if (T_1, S_1, t_2) is changed from ‘5’ to ‘2’ then entry corresponding to (T_1, S_3, t_2) is made zero. And also entries corresponding to (T_3, S_1, t_2), (T_4, S_1, t_2), (T_5, S_1, t_2), (T_7, S_1, t_2) and (T_9, S_1, t_2) are all made zeros. This time Matrix generated is as shown in the table given in table 3.

K varies	1 to 7	8 to 14	15 to 21	22 to 28	29 to 35	35 to 39	No of SP
(T_1, S_1)	0200000	0020000	0000020	0000200	0000000	0000	4
(T_1, S_3)	0000200	0000020	0020000	0200000	0000000	0000	4
(T_2, S_2)	2000000	0020000	0000020	0000200	0000000	0000	4
(T_2, S_3)	0002000	0000000	0000000	0000000	0000002	0000	2
(T_3, S_1)	0020000	0002000	0000200	0000002	0000000	0000	4
(T_3, S_3)	0000020	0000002	0000000	0002000	0000200	0000	4
(T_4, S_1)	2000000	0200000	0000000	0002000	0000002	0000	4
(T_4, S_2)	0002000	0000200	0020000	0000002	0000000	0000	4
(T_5, S_1)	0000000	0000020	2000000	0200000	0000200	0000	4
(T_5, S_3)	0000000	0020000	0002000	0000200	0200000	0000	4
(T_6, S_2)	0000200	2000000	0000002	0000000	0002000	0000	4
(T_6, S_3)	0200000	0002000	0000000	0020000	0000000	0000	3
(T_7, S_1)	0000002	0000000	0000002	0000000	0000000	0200	3
(T_7, S_2)	0000000	0000002	0000000	0020000	0200000	0000	3
(T_8, S_2)	0200000	0000020	0000000	0000000	2000000	0020	4
(T_8, S_3)	0000000	0000000	0000000	0000000	0002000	0000	1
(T_9, S_1)	0000000	0000000	0000000	0000020	0000000	0000	1
(T_9, S_2)	0000000	0000000	0002000	0000000	0000000	0000	1

TABLE 3. Selected periods are replaced by ‘2’

If the number of selected periods in any of the rows is not equal to 4**,** then replace ‘h’ as well as ‘m’ by ‘4’ in such a row**.** For the above example, in 4th, 12th, 13th, 14th, 16th, 17th and 18th row, the number of selected periods are 2, 3, 3, 3, 1, 1 and 1 respectively. Hence in those rows, both ‘h’ and ‘m’ are replaced by ‘4’ as shown in the table given below.

K varies	1 to 7	8 to 14	15 to 21	22 to 28	29 to 35	35 to 39
(T_1, S_1)	0500000	0050000	0000050	0000500	5000005	0005
(T_1, S_3)	0000500	0000050	0050000	0500000	0005000	5000
(T_2, S_2)	5000005	0050000	0000050	0000500	0005000	0500
(T_2, S_3)	0044400	4000440	0044000	4440004	0400044	0004
(T_3, S_1)	0050000	0005000	0000500	0000005	0500000	5000
(T_3, S_3)	0000050	0000005	0500000	0005000	0000500	0005
(T_4, S_1)	5000005	0500000	0000050	0005000	0000005	0500
(T_4, S_2)	0005000	0000500	0050000	0000005	0005000	0000
(T_5, S_1)	0050000	0000050	5000005	0500000	0000500	0005
(T_5, S_3)	0000050	0050000	0005000	0000500	0500000	5000
(T_6, S_2)	0000500	5000005	0000005	0000050	0005000	0050
(T_6, S_3)	0440044	0044400	0004400	0044000	4400044	4000
(T_7, S_1)	0400044	0004400	4400044	4000440	0004400	0400
(T_7, S_2)	0004400	0400044	0044400	0044000	4440004	0004
(T_8, S_2)	0500000	0000050	0000005	0000500	5000005	0050
(T_8, S_3)	0004400	0044000	4004400	0440044	0044400	4000
(T_9, S_1)	4400044	0044000	4400044	4000440	0004400	0400
(T_9, S_2)	0044400	4000440	0004400	0044000	4440004	0004

TABLE 4: ‘h’ and ‘m’ are replaced by ‘4’ in rows where SP not equal to 4

Once again generate matrix of selected periods which satisfy the constraints in section 2 by selecting and replacing 5s and 4s by 2s as shown in the table given below.

K Varies	1 to 7	8 to 14	15 to 21	22 to 28	29 to 35	35 to 39	No of SP
(T_1, S_1)	0200000	0020000	0000020	0000200	0000000	0000	4
(T_1, S_3)	0000200	0000020	0020000	0200000	0000000	0000	4
(T_2, S_2)	2000000	0020000	0000020	0000200	0000000	0000	4
(T_2, S_3)	0020000	0000200	0002000	2000000	0000000	0000	4
(T_3, S_1)	0020000	0002000	0000200	0000002	0000000	0000	4
(T_3, S_3)	0000020	0000002	0000000	0002000	0000200	0000	4
(T_4, S_1)	2000000	0200000	0000000	0002000	0000002	0000	4
(T_4, S_2)	0002000	0000200	0020000	0000002	0000000	0000	4
(T_5, S_1)	0000000	0000020	2000000	0200000	0000200	0000	4
(T_5, S_3)	0000000	0020000	0000000	0000200	0200000	2000	4
(T_6, S_2)	0000200	0000002	0000000	0000020	0002000	0000	4
(T_6, S_3)	0200000	0002000	0000200	0020000	0000000	0000	4
(T_7, S_1)	0000020	0000200	0200000	0000000	0002000	0000	4
(T_7, S_2)	0000000	0200000	0002000	0020000	2000000	0000	4
(T_8, S_2)	0200000	0000020	0000002	0000000	0000002	0000	4
(T_8, S_3)	0002000	0000000	2000000	0000020	0020000	0000	4
(T_9, S_1)	0000002	0000000	0000002	0000000	0000000	0200	3
(T_9, S_2)	0020000	0000000	0000200	0002000	0000000	0002	4

TABLE 5: Selected periods are replaced by ‘2’

If the number of selected periods in any of the rows is not equal to four, then once again replace ‘h’, ’m’ as well as ’l’ by “3” in such a row. For the above example, the number of selected periods is three in 17th row. Hence in that row h’, ’m’ and ’l’ are replaced by “3”. This is illustrated in the table given below.

K varies	1 to 7	8 to 14	15 to 21	22 to 28	29 to 35	35 to 39
(T_1, S_1)	0550005	0055000	5000550	0005500	5500055	0055
(T_1, S_3)	0000500	0000050	0050000	0500000	0005000	5000
(T_2, S_2)	5000005	0050000	0000050	0000500	0005000	0500
(T_2, S_3)	0044400	4000440	0044000	4440004	0400044	0004
(T_3, S_1)	0050000	0005000	0000500	0000005	0500000	5000
(T_3, S_3)	0000050	0000005	0500000	0005000	0000500	0005
(T_4, S_1)	5000005	0500000	0000050	0005000	0000005	0500
(T_4, S_2)	0005000	0000500	0050000	0000005	0005000	0000
(T_5, S_1)	0050000	0000050	5000005	0500000	0000500	0005
(T_5, S_3)	0000050	0050000	0005000	0000500	0500000	5000
(T_6, S_2)	0000500	5000005	0000005	0000050	0005000	0050
(T_6, S_3)	0440044	0044400	0004400	0044000	4400044	4000
(T_7, S_1)	0400044	0004400	4400044	4000440	0004400	0400
(T_7, S_2)	0004400	0400044	0044400	0044000	4440004	0004
(T_8, S_2)	0500000	0000050	0000005	0000500	5000005	0050
(T_8, S_3)	0004400	0044000	4004400	0440044	0044400	4000
(T_9, S_1)	3330333	0033303	3330333	3303330	3033300	0300
(T_9, S_2)	0044400	4000440	0004400	0044000	4440004	0004

TABLE 6. ‘h’ ‘m’ and ‘l’ are replaced by ‘3’ in rows where SP not equal to 4

Once again generate matrix of selected periods which satisfy the constraints in section 2 by selecting and replacing 5s, 4s and 3s by 2s as shown in the table given below.

K varies	**1 to 7**	**8 to 14**	**15 to 21**	**22 to 28**	**29 to 35**	**35 to39**	**No Of SP**
(T1,S1)	0200000	0020000	2000000	0002000	0000000	0000	4
(T1,S3)	0000200	0000020	0020000	0200000	0000000	0000	4
(T2,S2)	2000000	0020000	0000020	0000200	0000000	0000	4
(T2,S3)	0020000	0000200	0002000	2000000	0000000	0000	4
(T3,S1)	0020000	0002000	0000200	0000002	0000000	0000	4
(T3,S3)	0000020	0000002	0000000	0002000	0000200	0000	4
(T4,S1)	2000000	0200000	0000020	0000000	0000002	0000	4
(T4,S2)	0002000	0000200	0020000	0000002	0000000	0000	4
(T5,S1)	0000000	0000020	0000002	0000000	0000200	0002	4
(T5,S3)	0000000	0020000	0000000	0000200	0200000	2000	4
(T6,S2)	0000200	0000002	0000000	0000020	0002000	0000	4
(T6,S3)	0200000	0002000	0000200	0020000	0000000	0000	4
(T7,S1)	0000020	0000200	0200000	2000000	0000000	0000	4
(T7,S2)	0000000	0200000	0002000	0020000	2000000	0000	4
(T8,S2)	0200000	0000020	0000002	0000000	0000002	0000	4
(T8,S3)	0002000	0000000	2000000	0000020	0020000	0000	4
(T9,S1)	0000200	0000002	0000000	0200000	2000000	0000	4
(T9,S2)	0020000	0000000	0000200	0002000	0000000	0002	4

TABLE 7. Selected periods are replaced by '2'

The number of selected periods is equal to **four** in all the rows indicating that it is possible to generate Time table for the given set of priorities satisfying the constraints mentioned in section 1. Finally, the Time table is generated for all the three semesters by indicating the time slot assigned for each teacher. This is shown in the Table given below.

Semester 1

Days	ses1	ses2	ses3	ses4	ses5	ses6	ses7
Mon	T_4	T_1	T_3	–	T_9	T_7	–
Tue	–	T_4	T_1	T_3	T_7	T_5	T_9
Wed	T_1	T_7	–	–	T_3	T_4	T_5
Thurs	T_7	T_9	–	T_1	–	–	T_3
Fri	T_9	–	–	–	T_5	–	T_4
Sat	–	–	–	T_5			

Semester 2

Days	ses1	ses2	ses3	ses4	ses5	ses6	ses7
Mon	T_2	T_8	T_9	T_4	T_6	–	–
Tue	–	T_7	T_2	–	T_4	T_8	T_6
Wed	–	–	T_4	T_7	T_9	T_2	T_8
Thurs	–	–	T_7	T_9	T_2	T_6	T_4
Fri	T_7	–	T_9	T_6	–	–	T_8
Sat	–	–	–	–			

Semester 3

Days	ses1	ses2	ses3	ses4	ses5	ses6	ses7
Mon	–	T_6	T_2	T_8	T_1	T_3	–
Tue	–	–	T_5	T_6	T_2	T_1	T_3
Wed	T_8	–	T_1	T_2	T_6	–	–
Thurs	T_2	T_1	T_6	T_3	T_5	T_8	–
Fri	–	T_5	T_8	–	T_3	–	–
Sat	T_5	–	–	–			

TABLE 8: Final Time table generated for example 2.1

3. CONCLUSION

Time table generated using Algorithmic approach always ensures a gap of at least one hour between two classes for a teacher in a day. All other constraints like four hours for each subject in a week, only one first hour slot for a teacher for a given subject in a week and at most one hour class in a day for a teacher for a given subject etc can be implemented by specifying proper conditions in the program.

4. REFERENCES

[1] Schaerf, "A Survey of Automatic Timetabling, Artificial Intelligence Review", Springer Netherlands, vol.13, no. 2, P 87 – 127, 1999.

[2] Nguyen Duc Thanh," Solving Timetabling Problem Using Genetic and Heuristic Algorithms", Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, P 472 – 477, IEEE 2007,

[3] Ender Ozcan and Ersan Ersoy, Final Exam Scheduler – FES, P 1356 – 1363, IEEE, 2005.

[4] L. A. Zadeh, "Fuzzy sets", Information and control 8, P 338 – 353, 1965.

Reliability Improvement in Logic Circuit Stochastic Computation

Jeremy M. Lakes jlakes@ou.edu
Electrical and Computer Engineering
University of Oklahoma
Norman, OK 73019 USA

Dr. Samuel C. Lee samlee@ou.edu
Electrical and Computer Engineering
University of Oklahoma
Norman, OK 73019 USA

Abstract

Defects and faults arise from physical imperfections and noise susceptibility of the analog circuit components used to create digital circuits resulting in computational errors. A probabilistic computational model is needed to quantify and analyze the effect of noisy signals on computational accuracy in digital circuits. This model computes the reliability of digital circuits meaning that the inputs and outputs and their implemented logic function need to be calculated probabilistically. The purpose of this paper is to present a new architecture for designing noise-tolerant digital circuits. The approach we propose is to use a class of single-input, single-output circuits called Reliability Enhancement Network Chain (RENC). A RENC is a concatenation of n simple logic circuits called Reliability Enhancement Network (REN). Each REN can increase the reliability of a digital circuit to a higher level. Reliability of the circuit can approach any desirable level when a RENC composed of a sufficient number of RENs is employed. Moreover, the proposed approach is applicable to the design of any logic circuit implemented with any logic technology.

Keywords: Digital Design, Fault Tolerance, Reliability, Stochastic Model, Probabilistic Model and Noisy Signals.

1. INTRODUCTION

One of the pioneers of reliable circuit design was J. von Neumann in 1952[1]. Von Neumann created a technique called multiplexing that is capable of designing reliable computing units using only unreliable devices[1]. This technique consists of two stages called the executive stage and restorative stage. The executive stage replaces a processing unit with N multiplexed processing units that have N copies of each input and N copies of each output of the unit. The outputs from the executive stage are then fed into the restorative stage which stifles the errors present in the executive stage. The most popular example of this is Von Neumann's NAND multiplexing. With the assumption that NAND gates were not reliable and that they failed at a constant rate, Von Neumann discovered that he could treat a bundle of unreliable gates as an ideal reliable gate if gate failures were sufficiently small and independent. He calculated an upper bound of 0.0107 as the maximum tolerable rate of gate failure for his method to work. However, in 1998 W. Evans and N. Pippenger found that the upper bound for failure of each NAND gate is about .08856 [2]. This multiplexing method is effective at suppressing both permanent and transient faults due to the large amounts of redundancy used.

Von Neumann's research was furthered in 1977 when a group showed that logarithmic redundancy is sufficient to implement most Boolean functions[3]. Later that same year they also proved that some Boolean functions require at least logarithmic redundancy [4]. Almost a decade later Nicholas Pippenger proved that a Boolean function with O(n) noiseless gates can be reliably computed using only O(c) gates [5], [6]. This means, of course, that the number of additional noisy gates needed to reliably implement a Boolean function differs only by a multiplicative constant. Then in 1988, Pippenger showed that the fault probability of each gate must be below .5 for von Neumann's method to work [7]. Additionally, he explained that more layers of redundancy are needed to compute reliably in a noisy environment [7]. Overall, R. L. Dobrushin, S. I. Ortyukov and N. Pippenger strengthened von Neumann's theories by providing changes and rigorous proofs to his work.

Much more recently groups have furthered von Neumann's multiplexing method by using Probabilistic Model Checking (PMC) to calculate the redundancy-reliability tradeoffs when designing digital circuits [8], [9]. PMC is an algorithmic procedure that determines if a given stochastic system satisfies probabilistic specifications. They have developed a defect-tolerant CAD framework, called NANOPRISM, using PMC so that optimized systems can be simulated with a desired level of reliability built into the circuit [8]. Then they furthered their research by showing that NAND multiplexing can achieve higher reliability at lower redundancy rates than what von Neumann originally proposed [9]. This system is capable of taking an entire Boolean network as input and to evaluate the reliability-redundancy tradeoffs so that a circuit designer can decide based on the cost of reliability. The research given in [8] and [9] is focused on the study of correcting reliability issues stemming from both permanent and transient faults.

Other groups have recently reexamined other types of reliability improvement for digital circuits [10], [11]. They explore the statistical characteristics of R-Fold modular redundancy and reconfigurable computing as options for improving reliability. Additionally, Han and Jonker discovered a system architecture based on NAND multiplexing that can increase reliability of a circuit to a satisfactory level [10]. This method, however, is only viable for ultra large scale integrations due to the amount of redundancy needed. They also discuss the reliability improvement technique of reconfigurable computing and discover that it requires even more redundancy than NAND multiplexing. Finally, Nikolik et. al. explores the possibility of using R-Fold modular redundancy [11]. This technique uses a number, R, of redundant processing units that have their outputs fed into a majority gate, where R=3, 5, 7, 9, ···. The majority gate will then output the value of the most frequently calculated Boolean value and chooses that as the final output. Han and Jonker show statistically that R-Fold modular redundancy and NAND-multiplexing are generally less effective than reconfiguration for protecting against manufacturing errors [10]. However, reconfiguration is virtually useless for protecting against transient errors. A comprehensive survey of techniques and architectures for providing defect and fault tolerance to digital circuits can be found in chapter 10 of a recently published book [12].

All the groups mentioned above have studied and provided solutions to the reliability problem in circuits caused by transient and permanent faults in computational circuits. However, no group has provided a solution to this reliability issue by focusing on correcting faults due primarily to noisy input signals. In this paper we present a design approach that can nullify the effects of temporary and permanent noise of any digital circuit. Our proposed approach provides a noise-tolerant architecture for designing reliable digital circuits that operate in noisy environments. The paper is organized as follows:
Section 2 introduces the stochastic/probabilistic behavior of digital circuits. The study of the probabilistic behavior of a 1-bit half adder in the stochastic domain reveals that the reliability of digital circuits with noisy signals is generally low. Section 3 discusses a class of digital circuits, called Reliability Enhancement Networks (REN) that can enhance reliability of digital circuits. A single REN can increase the reliability of a digital circuit, but greater reliability can be achieved when multiple RENs are cascaded in series to form Reliability Enhancement Network Chains (RENC) which is also presented in Section 3. Section 4 shows a software simulation of RENC in action employed on a 1-bit half adder and then a conclusion is delivered.

2. PROBABILISTIC BEHAVIOR OF DIGITAL CIRCUITS IN A NOISY ENVIRONMENT

Since the advent of digital circuitry not only the size of circuits has been reduced, but also the design approach has changed drastically[12], [13]. All input and output signals of noisy digital circuits are considered to be random variables due to the high level of noise bombarding input signals which creates a high signal to noise ratio[15], [16]. Therefore, a probabilistic model is needed to calculate the outputs of digital circuits by using probabilistic inputs and outputs that describe the likelihood of each signal being a logic one [17], [18]. This stochastic model takes Boolean functions of noisy digital circuits and replaces them with equivalent probability functions [19], [20]. Figure 1 presents a model of digital logic gates with noise applied to the inputs. Note that computing with a probabilistic model is also called stochastic computing[18]; the two terms will be used interchangeably throughout this paper. It should also be noted that when the signal to noise ratio of a digital circuit is kept below a certain threshold, the circuit can still perform normally. Obviously creating a noise free environment for all digital circuits to operate within is not possible so reliable digital circuits are still a challenge to produce.

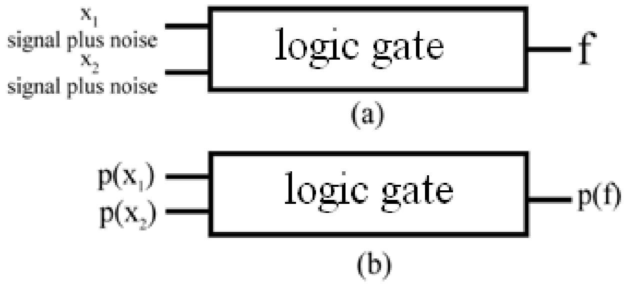


FIGURE 1:(a) Noisy model of a logic gate (b) Probabilistic/stochastic computing model where p(x) is the probability of that signal being a 1.

Sources of noise in digital circuits include thermodynamic fluctuations, electromagnetic interference, radiation, and parameter fluctuations. The noise can affect the timing, causing a delay failure, increase power consumption, and cause function failure because of signal deviation. The probabilistic behavior of digital circuits in a noisy environment is studied using the stochastic computing model where the input and output signals are described as probabilities of being logic 1 or logic 0.

Let the inputs of the AND gate $x_1, x_2 \in [0,1]$ be mutually independent with probabilities represented by $p_1 = p(x_1)$ and $p_2 = p(x_2)$. The output probability $p = p(f)$ can be evaluated using the probability of both events x_1 and x_2, i.e., $p = p_1 \cdot p_2$. Now consider the OR gate with inputs and outputs labeled the same as our AND gate. The output probability of this OR gate is $p = p_1 + p_2 - p_1 \cdot p_2$. The output probability of a NOT gate is $p = 1 - p_1$. Note that if $p_1 = p_2 = 1$, the inputs become deterministic and the output for the AND, OR, and NOT gate become 1, 1, and 0, respectively, as expected. Also, note that with p's replaced by x's, i.e., AND: $= x_1 \cdot x_2$; OR: $f = x_1 + x_2 - x_1 \cdot x_2$; and NOT: $f = 1 - x_1$, the three output probability expressions become their respective arithmetic expressions. Similarly, the output probability of the XOR gate is $p = p_1(1 - p_2) + p_2(1 - p_1) = p_1 + p_2 - 2 \cdot p_1 \cdot p_2$ and the arithmetic expression of the output of the XOR gate is $f = x_1 + x_2 - 2 \cdot x_1 \cdot x_2$. The output probability of NAND, NOR, and XNOR are 1 minus that of AND, OR, and XOR, respectively. The above equations are listed in Table 1.

Type of gate	Output probability
AND	$p_1 \cdot p_2$
OR	$p_1 + p_2 - p_1 \cdot p_2$
NOT	$1 - p_1$
XOR	$p_1 + p_2 - 2 \cdot p_1 \cdot p_2$
NAND	$1 - p_1 \cdot p_2$
NOR	$(1 - p_1)(1 - p_2)$
XNOR	$1 - (p_1 + p_2 - 2 \cdot p_1 \cdot p_2)$

TABLE 1: Probabilistic Behavior of Digital Gates

Consider the 1-bit binary half adder circuit shown in Figure 2. The logic function for the sum S is $S = A \oplus B$ and the logic function for the carry out C is $C = A \cdot B$. The arithmetic expressions (transformations) of these two functions are $p_S = p_A + p_B - 2p_A p_B$ and $p_C = p_A p_B$ where

$p_A = p(A), p_B = p(B), p_S = p(S)$and $p_C = p(C)$. For example, when $p_A = p_B = 0.9$ then p_S and p_C are found to be $p_S = 0.18$and $p_C = 0.81$. This means that the output S has a 0.82 chance of being a 0 and the output probability of C dictates that it has a 0.81 chance of being a 1. In other words, each output has a fairly high chance (nearly 20%) of delivering an incorrect answer when noise is accommodated.

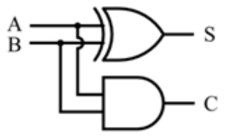


FIGURE 2: 1-bit binary half adder circuit

3. PROPOSED NEW METHOD

The equations given in this section can be used to calculate the required length of the RENC to meet reliability specifications of a noisy digital circuit. Furthermore, RENC can be applied to any number of outputs with either the same or varying reliability requirements at each output.

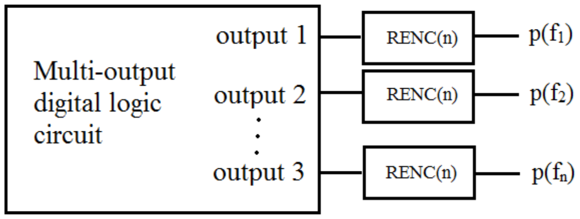


FIGURE 3: Multi-output digital circuit reliability improvements where the value n signifies the number of RENs used to construct the RENC.

This approach is a new way of looking at the reliability enhancement of digital circuits because it is focused on correcting the effect of the high signal to noise ratio in digital circuits. Digital circuits of any kind could be much more commercially viable if this high signal to noise ratio problem is remedied with our novel approach. A class of REN digital circuits, which are the building blocks of RENC, is presented in the following section.

3.1 A Class of REN Digital Circuits

A new approach for designing noise-tolerant digital circuits is proposed. For this approach to be viable we show that some real circuits exist that satisfy all the requirements of an REN. Our first example of an REN is illustrated in Figure 4. This REN, labeled REN1, has a probability function that can be derived by using stochastic model introduced in Section 2.

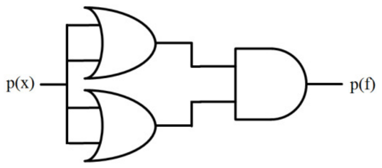


FIGURE 4: Reliability Enhancement Network 1 (REN1)

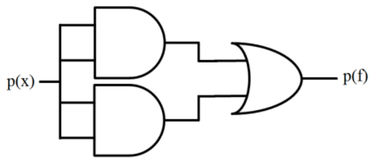


FIGURE 5: Reliability Enhancement Network 2 (REN2)

REN1s probability function is derived using Table 1 and can be expressed as $F(X) = (2X - X^2)^2$, where $F(X) = p(f)\varepsilon[0.1]$ and $X = p(x)\varepsilon[0.1]$. Figure 6a gives the plot of REN1s probability function to visually verify the two required properties from Definition 1. Examining the probability function of REN1 closely reveals that the output will be lower than the input when the input to REN1 is lower than X_p. Likewise, the output will be higher than the input when the input to REN1 is higher than X_p. For example, $F(0.2) = 0.1296$ and $F(0.9) = 0.9801$ meaning that REN1 has the ability to improve circuit output reliability with the stochastic threshold being $X_p = 0.381966$. There is one other REN shown in Figure 6b. The plot, probability function and pivot point are shown in Figure 6b and Table 2. It is shown that they are all RENs and can be used to construct RENC for enhancing reliability of digital circuits.

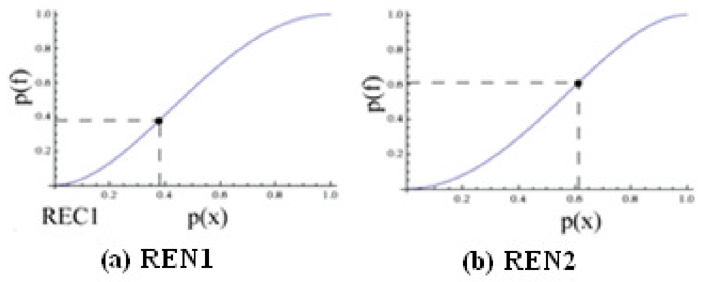


FIGURE 6: Plot of the probability function of REN1 and REN2

REN Name	Function	Pivot Point
REN1	$F(X) = (2X-X^2)^2$	$X_p \approx 0.381966$
REN2	$F(X) = 2X^2-X^4$	$X_p \approx 0.618030$

TABLE 2: Probability functions and X_p values for presented RENs

Given just these two RENs we now have the capability to construct some useful RENC for the improvement of output reliability of digital circuits. This will be further explained in the next section.

3.2 Reliability Enhancement Network Chains

The reliability enhancement capabilities of a single REN are not generally enough to make a noisy digital circuit reliable. In this case, RENCs are needed and can be constructed by cascading multiple RENs at an output.

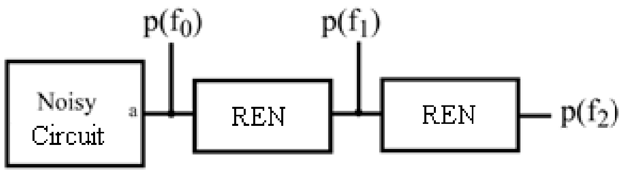


FIGURE 7: Noise-tolerant configuration using RENC

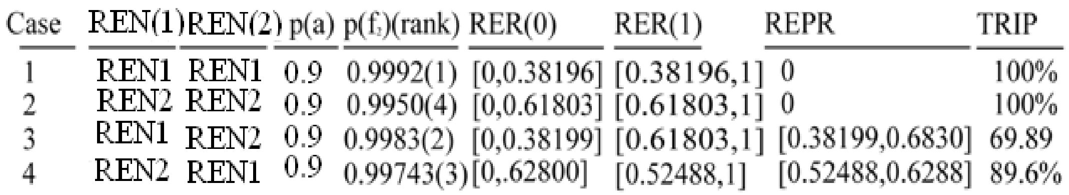
Consider a noisy single output digital circuit whose output a is connected to a 2-cell RENC, illustrated in Figure 7. Let us consider the following four cases: (1) both cells are REN1 of Figure 4, (2) both cells are REN2 of Figure 5, (3) a REN1 circuit followed by a REN2 circuit and (4) a REN2 circuit followed by a REN1 circuit. The first two RENCs will be referred to as homogenous RENC or HRENC and the last two RENCs will be referred to as non-homogenous RENC or NHRENC.

Definition 1
Reliability Enhancement Ranges for logic 0 (RER(0)) and logic 1 (RER(1)) are the ranges where the stochastic logic 0 and logic 1 can be enhanced by REN or RENC.

Definition 2
Reliability Enhancement Prohibited Range (REPR) is defined as the range of X where the reliability improvement of an RENC is unpredictable. This range lies between the X_ps of different RENs in an RENC. This happens when the RENs used to create an RENC are not of the same type.

Definition 3
The Total Reliability Improvement Percentage (TRIP) is the percentage of X values where reliability of a circuit can be enhanced though a REN or RENC. That is, TRIP=1-REPR.

To compare their various reliability enhancement capabilities, we arbitrarily choose $p(a) = 0.9$ as an example. The $p(f_2), RER(0), RER(1), REPR\ and\ TRIP$ of the four 2-cell RENCs are tabulated in Table 1.



Case	REN(1)	REN(2)	p(a)	p(f$_2$)(rank)	RER(0)	RER(1)	REPR	TRIP
1	REN1	REN1	0.9	0.9992(1)	[0,0.38196]	[0.38196,1]	0	100%
2	REN2	REN2	0.9	0.9950(4)	[0,0.61803]	[0.61803,1]	0	100%
3	REN1	REN2	0.9	0.9983(2)	[0,0.38199]	[0.61803,1]	[0.38199,0.6830]	69.89
4	REN2	REN1	0.9	0.99743(3)	[0,.62800]	[0.52488,1]	[0.52488,0.6288]	89.6%

TABLE 3: Tabulations of different HRENC and NHRENC configurations

From $p(f_2)$ in Table 3, we see that HRENC and NHRENC all can deliver desirable reliability. However, from the computational point of view it is much easier to determine the number of stages needed for a HRENC to meet a required level of reliability. A design example of a noise tolerant digital circuit using RENC through simulation is presented in the next section.

4. RELIABILITY ENHANCEMENT NETWORK CHAIN SIMULATION

In this section we present simulations of the design examples given in the previous section created in MATLAB. The circuit that we simulated is a half adder. This simulation is a two-variable plot with half adder inputs $A \in [0,1]$ and $B \in [0,1]$ used to derive the output $S \in [0,1]$ through the probability function of S which is illustrated in Figure 8a. This plot shows the output S without any special improvement. This digital circuit clearly needs reliability improvement; this can be seen in how most points on this graph are not near logic 1 or logic 0.

The second graph (Figure 8b) shows what the final output S looks like after just one REN1 is attached. The improvement is a bit more clear after 2 REN1 stages because there is much more area on the graph that is very close to either logic 1 or logic 0 as shown in Figure 8c. Improvement is very obvious, as shown in Figure 8d, when we simulate five REN1 stages after the output S because the amount of graph area very close to logic 1 or logic 0 is greatly increased and the transition between the two is much sharper. This illustrates how the output of a stochastic circuit can be made close to deterministic when a well-designed RENC is employed.

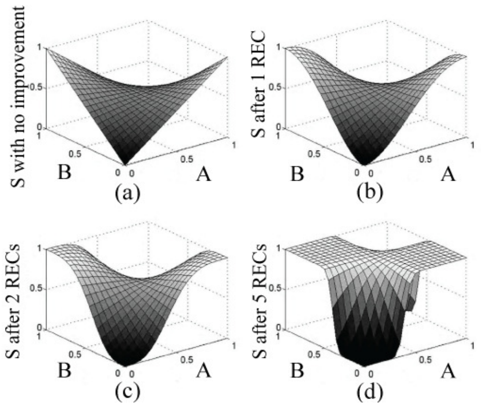


FIGURE 8: Improvement of S with RENC

5. COMPARATIVE EVALUATION OF THE RESULTS

5.1 Comparative Example

Consider a comparative example using the 1-bit binary half adder (HA). Referring to figure 2, the probability functions of the 1-bit HA are $p_S = p_A + p_B - 2p_A p_B$ and $p_{Cout} = p_A p_B$. Assume that the probabilistic inputs to A and B are $p_A = p_B = 0.99$. Then $p_S = 0.0198$ and $p_{Cout} = 0.9801$. The probability of a logic 1 for the S output can be converted to reliability by subtracting it from 1 resulting in $p_S = 0.9802$. Now assume that the design required reliability for both p_S and p_{Cout} needs to be at least 0.99999.

RENC Method:

Suppose that REC1 is to be used, whose probability function is $F(X) = (2X - X^2)^2$. The reliability of the S and C_{out} outputs without improvement are $p_S = 0.9802$ and $p_{Cout} = 0.9801$. These reliabilities are increased to $p_S = 0.9984627359$ and $p_{Cout} = 0.9992081368$ with the addition of one REC1 at each output. The S and C_{out} outputs do not meet the reliability requirements by attaching one REC1 so we must add one more REC1 to each RENC. This increases the reliability of each output to $p_S = 0.9999905618$ and $p_{Cout} = 0.9999987459$ which are greater than the required reliability.This means that means we need at least an RENC of two REC1s at each output of the HA. The reliabilities of p_S and p_{Cout} are tabulated in table 4.

Number of REC1s	0	1	2
p_S	0.9802	0.9984627359	0.9999905618
p_{Cout}	0.9801	0.9992081368	0.9999987459

TABLE 4: Tabulations of RENC for 1-bit HA

R-Fold Modular Redundancy (RFMR):

To meet the same design reliability requirements of p_S and p_{Cout} being at least 0.99999 using RFMR the output reliabilities p_S and p_{Cout} are tabulated for R=3, 5 and 7using the binomial formula: $B(r; n, p) = \binom{n}{r} p^r (1-p)^{n-r}$. The reliability of 3MR is $R_{3MR} = 3p^2 - 2p^3$. We find that the reliability for the S output is $R_{3MR}(S) = 0.9988394048$ when $p_S = 0.9802$ and the reliability for the C_{out} output is $R_{3MR}(C_{out}) = 0.9988277312$ when $p_{Cout} = 0.9801$. These reliabilities are not as high as the design specifies so we will move on to R=5 (5MR). Using the binomial theorem again we find the reliability function of 5MR is $R_{5MR} = 6p^5 - 15p^4 + 10p^3$. We find that the reliability for the S output is $R_{5MR}(S) = 0.9999246633$ and the reliability for C_{out} becomes $R_{5MR}(C_{out}) = 0.9999235276$. These output reliabilities are still not as high as the design specifies so we must try R=7 (7MR). Repeating the same process we find that the reliability function of 7MR is $R_{7MR} = -20p^7 + 70p^6 - 84p^5 + 35p^4$. Now the reliabilities of the HA outputs become $R_{7MR}(S) = 0.9999948721$ and $R_{7MR}(C_{out}) = 0.999994769$ which both meet the reliability requirements of the design. A tabulation of the results is given in table 5.

R	3	5	7
p_S	0.9988394048	0.9999246633	0.9999948721
p_{Cout}	0.9988277312	0.9999235276	0.9999948721

TABLE 5: Tabulations of R-Fold Modular Redundancy on the 1 bit HA

5.2 Evaluation of the Results

Now that we have shown that both the RENC using two REC1s method and the 7MR can both meet our reliability requirements. The RENC method only requires one 1-bit HA and four REC1s. Since we know that each REC1 requires three logic gates and If we assume the HA requires two logic gates then the total gates required for this method is 14.

The 7MR circuit requires seven 1-bit HA and two 7-input Majority Gates.One way to estimate the number of gates to construct the 7-input Majority Gate is as follows. The 7-input Majority Gate requires a combinational circuit which is composed of 35 minterms. Each minterm has four variables, which means each of the 35 minterms uses three AND gates for a total of 105 AND gates. Additionally, 35 minterms requires 34 OR gates to create the sum of products for the 7-input majority gate for a total of 139 logic gates. The total number of gates required for this design is then calculated as $(139 \cdot 2) + (2 \cdot 7) = 292$.Table 6 has a tabulation of these results. Hence, for this example is it seen that the required number of gates for the RENC method compared to the RFMR method is approximately 1:20.

Reliability Enhancement Type	**Logic Gate Count**
RENC	14
R-Fold Modular Redundancy	292

TABLE 6: Tabulation of Logic Gate Count

6. CONCLUSION

This paper presents a new approach to the design of noise-tolerant digital circuits that utilizes Reliability Enhancement Network Chains. Our approach to the reliability problem of digital circuits is novel because it hinders environmental noise at the output of a computational unit rather than using redundant computational units like many traditional methods. This fact is very important because employing massive redundancy of computational units to create reliable digital circuits can negate the benefit of the high chip density allowed by modern digital technologies. In addition to providing higher chip density, our approach delivers high reliability in the face of random environmental signal noise by adding a small number of additional gates to the design. Furthermore, it has been shown that the reliability of digital circuits approaches deterministic levels as the number of REN stages of the RENC is sufficiently large.

7. REFERENCES

[1] J. von Neumann,"Probabilistic logics and the synthesis of reliable organisms from unreliable components," in Automata Studies, C.E. Shannon and J. McCarthy, Eds. Princeton, NJ:Princeton University Press, 1956, pp. 43-98.

[2] W. Evans and N. Pippenger."On the maximum tolerable noise for reliable computation by formulas" IEEE Trans. Inform. Theory, vol. 44, pp. 1299-1305, 1998.

[3] R. L. Dobrushin and S. I. Ortyukov."Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements" Prob. Inform. Trans., vol. 13, pp. 203-218, 1977.

[4] R. L. Dobrushin and S. I. Ortyukov."Lower bound on the redundancy of self-correcting arrangements of unreliable functional elements" Prob. Inform. Trans., vol. 13, pp. 59-65, 1977.

[5] N. Pippenger. "On networks of noisy gates" in Proc. 26thAnnu. Symp. Foundationas of Computer Science, pp. 30-38, 1985.

[6] N. Pippenger. “Invariance of complexity measures for networks with unreliable gates” J. ACM, vol 36, pp. 194-197, 1988.

[7] N. Pippenger. “Reliable computation by formulas in the presence of noise,” IEEE Trans. Inform. Theory, vol. 34, pp. 194-197, 1988.

[8] G. Norman, D. Parker, M. Kwiatkowska and S. Shukla.”Evaluating reliability of defect tolerant architecture for nanotechnology using probabilistic model checking” in Proc. IEEE VLSI Design Conference (IEEE Press, 2004), to appear.

[9] D. Bhaduri and S. K. Shukla. "Reliability evaluation of von Neumann multiplexing based defect-tolerant majority circuits" *Nanotechnology, 2004. 4th IEEE Conference on* , vol., no., pp. 599-601, 16-19 Aug. 2004.

[10] Jie Han and Pieter Jonker.“A System Architecture Solution for Unreliable Nanoelectronic Devices” IEEE Trans. on Nanotechnology, vol. 1, no. 4, pp. 201-208, December 2002.

[11] K. Nikolic, A. Sadek and M. Forshaw.”Architectures for Reliable Computing with Unreliable Nanpdevices” in Proc. IEEE-NANO ’01 (IEEE 2001), pp. 254-259.

[12] S. Ahuja, G. Singh, D. Bhaduri, and S. Shukla.”Fault- and Defect-Tolerant Architectures for Nanocomputing” in Bio-inspired and Nanoscale: Integrated Computing, M. Wilner, Ed. New Jersey: Wiley, 2009, pp. 262-293.

[13] X. Lu and J. Li.”Probabilistic Modeling of Nanoscale XOR Gate” in *International Conference on Apperceiving Computing and Intelligence Analysis*, 2008, pp. 4-7.

[14] X. Lu, J. Li, G. Yang, and X. Song.”Probabilistic Modeling of Nanoscale Adder” in *IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, 2008.

[15] S. C. Lee and L. R. Hook."Toward the design of a nanocomputer" in Proc. IEEE 2006 Canadian Conference on Electrical and Computer Engineering, May 2006.

[16] S. C. Lee and L. R. Hook. "Logic and Computer Design in Nanospace" IEEE Transactions on Computers, vol. 57, no. 7, pp. 965-977, 2008.

[17] T. Rejimon, K. Lingasurbramanian, and S. Bhanja.“Probabilistic Error Modeling for Nano-Domain Logic Circuits” IEEE Trans. On VLSI, vol. 17, no. 1, pp. 55-65, January 2009.

[18] W.R. Fahrner, Ed., *Nanotechnology and Nanoelectronics: Materials, Devices, Measurement Techniques*. Berlin: Springer, 2005, pp. 18-38.

[19] S. Yanushkevich, V. Shmerko, and S. Lyshevski.*Logic Design of NanoICs*. Boca Raton: CRC Press, 2005.

[20] S. Yanushkevich, V. Shmerko, and S. Lyshevski.*Computer Arithmetics for Nanoelectronics*. Boca Raton: CRC Press, 2009.

INSTRUCTIONS TO CONTRIBUTORS

The International Journal of Logic and Computation aims to promote the growth of logic and computing research from the perspectives of logic, mathematics and computer science, but emphasizes semantics of programs, in contrast with the traditional treatment of formal languages as sets of strings. IJLP promote this new field with its comprehensive selection of technical scientific papers and regular contributions such as letters, reviews and discussions for logical systems using classical and non-classical logic, constructive logic, categorical logic, modal logic, type theory, logical issues in logic programming, knowledge-based systems and automated reasoning programing; logical programming issues in knowledge representation, non-monotonic reasoning, logics and semantics of programming and applications of logic in hardware and VLSI.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJLP.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2 2011, IJLP appears in more focused issues. Besides normal publications, IJLP intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

IJLP LIST OF TOPICS

The realm of International Journal of Logic and Computation (IJLP) extends, but not limited, to the following:

- Categorical Logic
- Classical and Non-Classical Logic
- Constructive Logic
- Logic Representation Techniques
- Logical Programming Issues in Knowledge Representa
- Modal Logic
- Non-Monotonic Reasoning
- Programming Reasoning Test Collection
- Semantic Representation in Logic Programming
- Challenges in Natural Language and Reasoning
- Computer Logical Reasoning
- Knowledge-Based Systems and Automated Reasoning Pr
- Logical Issues in Logic Programming
- Logics and Semantics of Programming
- Natural Language
- Programming Expressiveness
- Reasoning Systems
- State-Based Semantics

CALL FOR PAPERS

Volume: 3 - Issue: 1 - February 2012

i. Paper Submission: November 30, 2011 **ii. Author Notification:** January 01, 2012

iii. Issue Publication: January / February 2012

CONTACT INFORMATION

CSC PUBLISHERS © 2011
COMPUTER SCIENCE JOURNALS SDN BHD
M-3-19, PLAZA DAMAS
SRI HARTAMAS
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6207 1607
006 03 2782 6991

FAX: 006 03 6207 1697
EMAIL: cscpress@cscjournals.org