International Journal of Advanced Science and Technology
Vol.80 (2015), pp.53-78
http://dx.doi.org/10.14257/ijast.2015.80.05

Cycle in a Conventional Combinational Circuit: A Comprehensive Survey

Alak Majumder[1], Bipasha Nath[2], Durba Sarkar[2] and Moushumi Das[2]

[1]*Assistant Professor, Department of ECE, National Institute of Technology Arunachal Pradesh*
[2]*Department of ETCE, Tripura Institute of Technology, Agartala India*
majumder.alak@gmail.com, nathbipasha.dmg@gmail.com

Abstract

A Boolean Combinational circuit can be made of some wires and logic gates whose outputs at any time are determined directly from the present combination of inputs without any regard to previous input. Reduction of number of gates (area) and the length of the signal path (delay) to optimize a Boolean circuit is always an overriding concern in the design of digital integrated circuit. The accepted phenomena that combinational circuit is based only on acyclic (loop-free or feed-forward) topology no longer exist. Introducing cycles in combinational circuit we may have same acyclic operation with reduced area and delay of the circuit. Cyclic circuits that do not hold state or oscillate are often the most convenient representation for certain functions, such as arbiters. The main aim of cyclic circuit is to introduce structural feedback & to avoid the logical feedback in order to get combinational primary output. In this survey, we advocate the technological depth of cyclic circuit and their design methodologies with certain merits and demerits. It includes functional analysis i.e. to determine what values will appear and the timing analysis which determine when these values will appear.

Keywords: *Combinational Circuit; Sequential Circuit; Cyclic Circuit; B-B Algorithm; DTTM Method; Karnaugh Map; Comparator*

1. Introduction

The world of digital circuits is widespread starting from the topic of semiconductor physics to the so called system level architecture. In the realm of this logic circuits the restriction in two valued signals *i.e.*, logic '0' and logic '1' brought its great essence over the world of digital electronics even. These logic values of digital system, however in practical can actually be a narrow band voltage which become advantageous over continuous values of voltages as in case of analogue system.

Broadly classified, Digital circuits are of two types –
(1) Combinational Circuit
(2) Sequential circuit

They can either be defined on the basis of behaviour as well as structure.

1.1. Behavioural Definition

1.1.1. Combinational Circuit: In combinational circuit, the output variables at any instant of time are dependent only on the present input variables.

ISSN: 2005-4238 IJAST

1.1.2. Sequential Circuit: In sequential circuit, the output variables at any instant of time depend not only on the present input variables but also on the past history of the system. Thus the sequential circuits have a memory unit to store information whereas combinational circuit do not.

1.2. Structural Definition

1.2.1. Combinational Circuit: It contains only feed-forward path. It can be said that it consists of an acyclic configuration of logic gates.

1.2.2. Sequential Circuit: It contains loop or feedback path *i.e.*, it consists of a cyclic configuration of logic gates and memory element.

The two types of logic circuits are illustrated with the given examples shown in the figures given below. Let us start with a combinational circuit.

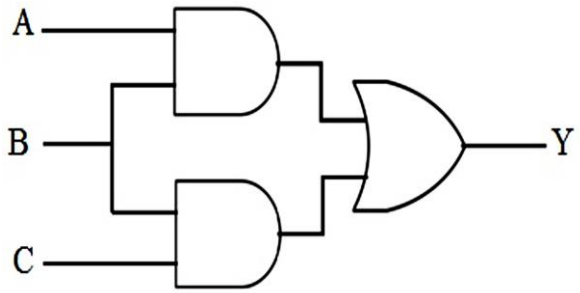


Figure 1. A Combinational Circuit

$$Y=b(c+ac^{/}) \quad (1)$$

Table 1. Truth Table of the Combinational Circuit

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

From the output equation we can say that the outputs are dependent only on present inputs. So it is called as combinational circuit.

Now let us observe the following circuit.

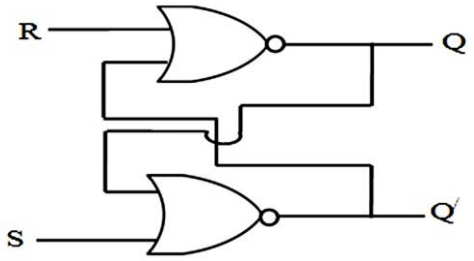


Figure 2. A SR Latch (a Sequential Circuit)

Table 2. Truth Table of the NOR- Gate Latch

R	S	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

The truth table clearly shows that the output function depends both on the present input variables as well as present state. Thus the circuit can be termed as sequential.

We were aware of the fact that combinational circuits are a collection of logic gates connected as a chain or branch like structure. They do possess acyclic topology *i.e.*, it contains only feed forward path. Regardless of the initial values on the wires, once the values of the inputs are fixed, the signals propagate to the outputs. There is a clear correspondence between the electrical behaviour of the circuit and the abstract notion of the Boolean functions that it implements. The behaviour of a circuit with feedback is generally more complicated. Such a circuit may exhibit timing-dependent behaviour (as in the case of an R-S Latch), and it may be unstable (as in the case of an oscillator). But indeed few examples will make us to realize that it may be possible to design a combinational circuit with feedback. Here we will corroborate about cycles in combinational circuits. Also we will demonstrate how this topology is helpful in optimization of combinational circuits in the fields of area and delay. Also we would discuss some analytical work approaches and processes for synthesizing such circuits that has been given by former researchers.

From the early era to till today a lot of researches contrived the fact that cyclic combinational circuits may exist. They did advocate the same thing that cyclic topology in combinational circuits reduces the circuit size.

2. Related Work

It is not very far away that directed logic gates became the important part of digital circuits. Almost 50 years ago, research communities started their work with logic gates. In the 1960's, research communities shifted themselves to logic gates (AND, OR, NOT *etc*). But before that also cyclic topologies were being used by digital circuits.

In 1938, Shannon first proposed analysis of switching circuits in his seminal paper [3]. His maximum examples include cyclic configurations. Relays are directionless, so cycles did not pose any problem those days.

In 1949, Shannon explained the synthesis of a switching circuit [4].

In 1953, Shannon described cyclic switching circuits with 18 contacts computes 16 Boolean functions of two variables [5]. He proved that his circuit is optimal.

In 1960, Short [11] gave an abstract on the graphical model to the study of switching circuit which put direction in switching circuits. He argued that cycles in combinational circuits are essential for minimal forms. He also suggested that in optimization of Binary Decision Diagram, feedback may be useful.

In 1963, McCaw presented a thesis named "Loops in Directed Combinational switching Network" for his Engineer's degree [6]. He explained that cycles in combinational can make a circuit to be smaller in size than an equivalent acyclic. McCaw stated that his circuit needed lesser gates than an equivalent acyclic circuit.

In 1970, Katz [7] published a short paper on feedbacks in logic circuits. He illustrated a cyclic circuit which had 6 fan-in 2 NOR gates with 3 inputs and outputs. According to rigorous model, his circuit was not combinational.

In 1971, Huffman [8] explained about feedback in linear threshold networks. He also described that cyclic is necessary for in the minimization of logic designs.

In 1977, Rivest [9] started with an intriguing example explaining that feedback path in combinational circuits make them smaller in size. Cyclic combinational circuits require lesser logic gates than an equivalent acyclic circuit.

In 1987, R.bryton, R.Rudell, A.Sangiovanni-Vincentelli and A. Wang had their paper published on the topic "MIS: A multi-level logic optimization system" [10] which leads to the study of cycles in combinational circuit.

In 1992, L. Stok presented a paper titled "False loop through resource sharing" [12]. In this paper he described the path in a cyclic combinational circuit. He told that there is lack of existence of logic and timing analysis programs in feedback of combinational circuits. In such examples, feedback is carefully contrived, occurring when functional units are connected in a cyclic topology.

In 1994, Sharad Malik [13] gave a paper explaining formal analysis of cyclic combinational circuits. In his paper he explained the techniques for the logical and timing analysis of such circuits based on ternary- valued.

In 1996, T.R Shiple, G.Berry and Touati proposed a Constructive analysis of cyclic circuit [14]. He extended Malik's work and set it on a firm theoretical footing. He showed that the class of circuits that Malik's procedure decides to be combinational are precisely those that are well-behaved electrically, according to the up-bounded inertial delay model. He proposed refinements to Malik's algorithm and extended the concept to combinational logic embedded in sequential circuits.

In 2004, Marc.D.Riedel, in his paper "Cyclic Combinational circuits" [1] described the general methodology for the synthesis of multilevel of combinational circuits with cyclic topology.

In 2004, Jie-Hong R. Jiang, Alan Mishchenko, and Robert k.Brayton proposed a paper titled "On brakable cyclic definitions" [15] where they focused on the analysis of combinationality in cyclic combinational circuit.

In 2006, O.Neiroukh, S.A. Edward and X.Song [16] proposed an algorithm to characterize exactly all combinational behaviour of a cyclic circuit and **in 2008** they shifted their work to transform a cyclic circuit to its equivalent acyclic one.

In 2010, Debopriyo Ghosh proposed a thesis named "Design of seven Segment Decoder using cyclic combinational Method and its implementation in FPGA"[18]. He designed the cyclic seven segment decoder of Marc D. Riedel approach. The

result of his simulation had shown that this cyclic implementation optimized power, delay and area.

3. Cycles in Combinational Circuits

Cycles in combinational circuits had first been devised by C. E. Shannon in the late 1930's [3]. In those days people studied switching circuits built from electro-mechanical relay. So he had relays as the building block of switching circuits as logic gates were the latter abstraction in the world of digital circuits. Relay is such a device that does not have any intrinsic direction. They can pass current in either direction unlike diodes. They do conduct current if it is "ON" (logic "0") and do not conduct if it is "OFF" (logic 1). It is also called a switch whose symbolic representation is just like an electric switch. Shannon was the first one who gave the symbolic logic to the switching circuits. Those days he used to have his maximum switching circuits with cyclic topology.

Consider the following diagram.

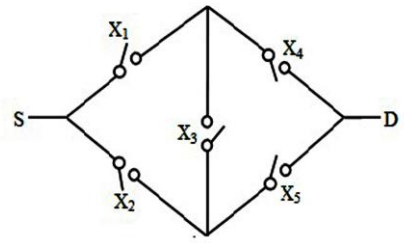


Figure 3. A Switching Circuit with Acyclic Topology

$$f(x_1, x_2, x_3, x_4, x_5) = x_1x_4 + x_1x_3x_5 + x_2x_5 + x_2x_3x_4 \quad (2)$$

This function has been implemented between points S and D. There is a direct connection between S and D. These kinds of circuit did not pose any problem regarding cycles in those circuits as relays are directionless. So neither the question of feedback and feed forward path and hence nor the question of combinational and sequential did arise.

Later, a quite good number of researchers contrived the fact that cyclic topology may be helpful in digital circuits in many fields. In the late 1960's digital world has been introduced with logic gates. Then digital circuits are divided into combinational and sequential as now we are aware of. But we studied that cycles can't occur in combinational circuits. If it is so, it would be called sequential. But many researchers proved the thing that it may be possible to design cyclic combinational circuits.

A simple example advocates the above statement [1].

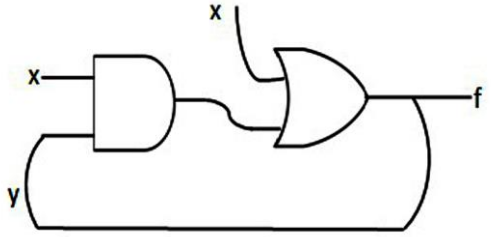


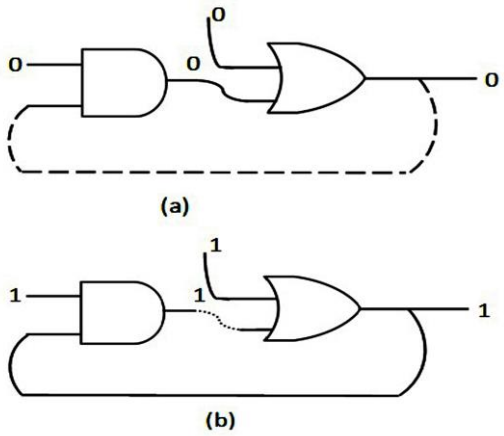
Figure 4. A Cyclic Combinational Circuit due to Marc. D Reidel

From the above figure, the output function can be derived easily. The function can be written as

$$f = x.\ y + x = x(y+1) = x \tag{3}$$

To be more precise, in the example given above,

If x = 0, the output of the AND gate is fixed at 0, and so the other input of AND from the output of OR gate has no influence.



On the other hand, if x = 1, the output of the OR gate is fixed at 1, and so the input from the AND gate has no influence. Although useless, this circuit is cyclic and combinational.

The value of the output f is determined by the current input value x (actually f = x) regardless of the prior state and independently of all timing assumptions.

Now comes the most intriguing example had ever given. Consider the following example [9]. This example will clearly prove that cycles may occur in combinational circuits.

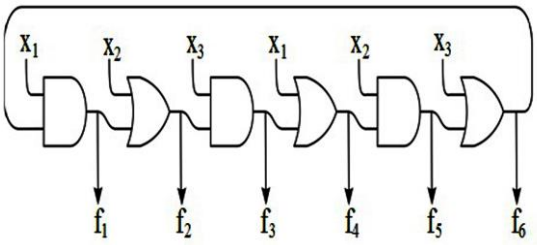


Figure 5. Cyclic Combinational Circuit by Rivest

Let us consider the feedback path in the above figure as y, that means $y = f_6$. The output functions are calculated as

$$f_1 = x_1 y \tag{4}$$

$$f_2 = x_2 + f_1 = x_2 + x_1 y \tag{5}$$

$$f_3 = x_3 f_2 = x_3(x_2 + x_1 y) \tag{6}$$

In 1963, McCaw [6] illustrated a cyclic circuit which consists of two AND gates and two OR gates with five inputs and two outputs. The circuit showing below corroborates that feedback in combinational circuits satisfying its combinationality is feasible.

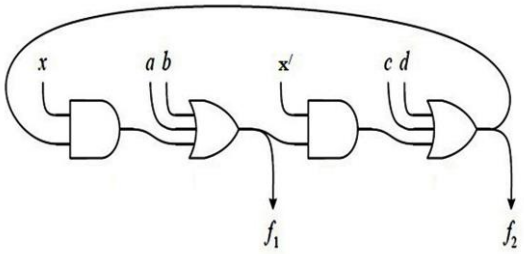


Figure 6. A Cyclic Combination Circuit due to McCaw

The outputs equations of the above circuit given by McCaw are given below

$$f_1 = a + b + x(c + d + x' f_1)$$
$$= a+b+x(c+d) \quad (7)$$

$$f_2 = c + d + x'(a + b + x f_2)$$
$$= c+d+x'(a+b) \quad (8)$$

The output equations advocate that the circuit is combinational as the output equations are independent of the feedback path. However, the circuit poses feedback in structural sense not in logical sense. We can conclude that circuits are combinational if all the cycles are false, the synthesized paths in the circuit never bite their own tail to form true cycles.

4. Logic Gate Reduction in Cyclic Environment

Cyclic combinational circuits are not only feasible to design but also advantageous. This can be proven by the following example given by Rivest.

Consider the following functions

$$f_1 = x_1(x_2 + x_3) \quad (9)$$

$$f_2 = x_2 + x_1 x_3 \quad (10)$$

$$f_3 = x_3(x_1 + x_2) \quad (11)$$

$$f_4 = x_1 + x_2 x_3 \quad (12)$$

$$f_5 = x_2(x_1 + x_3) \quad (13)$$

$$f_6 = x_3 + x_1 x_2 \quad (14)$$

If we are asked to implement the following functions using fan-in two gates, then we will be needed 12 gates as shown [2].

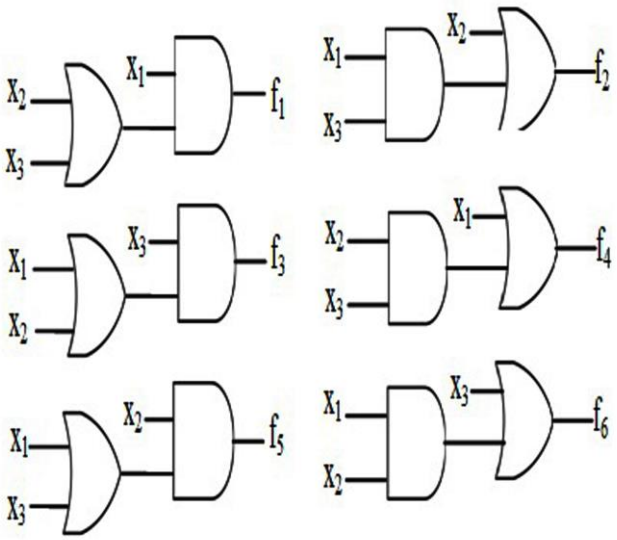


Figure 7. Gates Implementing 6 Functions

If we combine them altogether and make a single circuit of acyclic topology then at least we will be left with 7 fan-in two gates. This can be illustrated below.

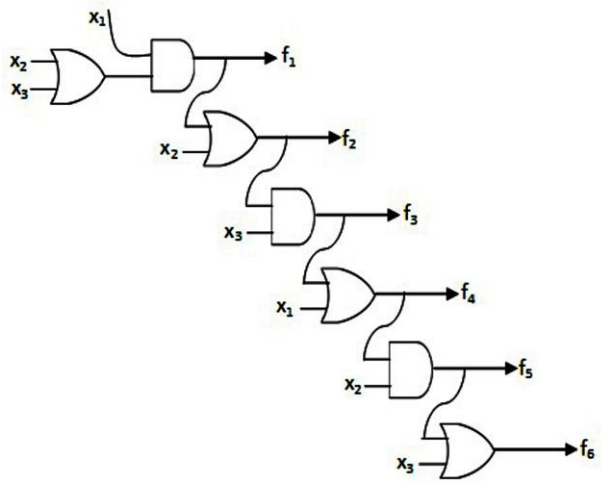


Figure 8. A Cyclic Construction of Equivalent Cyclic Circuit given by Rivest

All the 6 output functions of the above figure are same as that of the previous output functions. Here all the 6 output functions are satisfied.

Now if we are added with cyclic topology then we can omit one gate and can built it up with only 6 fan-in two gates as shown in figure 5. The circuit is combinational in true sense. In those circuits feedback is only in topological. Unlike sequential circuits, they do not act as logical feedback. So we can conclude that the cyclic topology in combinational circuits is very helpful in optimizing the circuit size. As gates are less so power will be less.

Migration to Cyclic circuit gives us an optimization (p)

$P = (12 – 6)/12 = 0.5 \times 100\% = 50\%$

But when we used cyclic combinational method the traversing path for the output function **f1** is reduced so the delay is reduced. For the output function **f2** the traversing path is same as in acyclic method. So for **f2** the delay is same. But from the output function **f3** to **f6** the traversing path & so the delay is gradually increases. Hence for this particular example, although total no of logic gates & power is less in case of cyclic combinational method but the overall delay of the logic circuit is more than that of the acyclic method.

5. Prevailing Cyclic Circuit

Presently, practitioners have happened to see that cycles occur in some combinational circuits synthesized from high level descriptions. But it has been observed that feedback may be inadvertent or may be carefully contrived. In these circuits, there is explicit "control" circuitry governing the interaction between "functional" units.

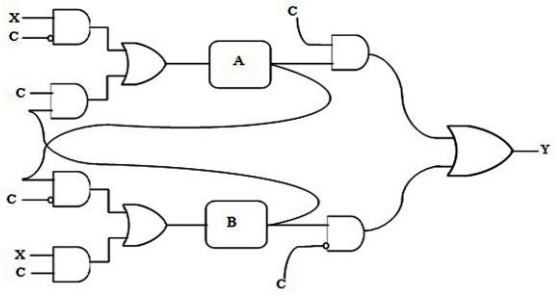


Figure 9. Functional Units Connected in Cyclic Topology

The following example clarifies the thing. In this figure there is an input word X which carries many bits of information and a control input c.

There are two functional units, A and B.

If C = 0

Then it computes B (A(X)).

If C = 1,

Then it is A (B(X)).

Suppose that $X=(x_1,..., x_n)$ is an n-bit word, representing the integer

$X_1+2X_2+.....+2^{n-1}X_n$

Here B(X) might be an exponentiation

$B(X) = 2^x \bmod 2^n$

And A(X) might be a left- shift (division by 2)

A(X) =X/2

The output equation will be expressed as

$$Y = C.A(B(X)) + C'.B(A(X)) \quad (15)$$

The circuit either performs a left-shift followed by an exponentiation, or an exponentiation followed by a left shift.

6. Theory of Cyclic Circuit

Generally, here, cyclic circuit refers to the combinational circuits with false feedback paths. Introduction of cycles in Boolean circuits are not only feasible but also advantageous. It helped in optimizing a Boolean circuit by reducing the size and delay.

We would analyze their behaviour, framework, underpinning circuit model etc to be more precise about them. Also we enter into the technological depth, design methodologies *etc*.

To do so, we need to have a glance over basic digital circuits.

Now, we explain the underpinning circuit model of any digital circuit and their analysis framework that characterises the functional behaviour.

6.1. Circuit Model

Digital circuits are differentiated with that of analog in the sense that it is provided with only two logic values i.e., logic '0' and logic '1'. The signals that are fed at the inputs are actually analog in nature. The wires recognises them as analog signals *i.e.*, continuous real valued function of the time y(t), corresponding to each voltage level.

However, it may happen that signals that arrived on the wires may not be either logic '1' or logic '0'. Rather, it may be an unknown value. In that case, analysis of any circuit becomes difficult. To overcome the problem, it is necessary to adopt ternary framework where the unknown value is represented as 'X'. Thus we proceed towards ternary extension. The logical values for corresponding analog signals may be given by the mapping [17]:

$$\text{Logical } [y(t)] = \begin{cases} 0 & \text{if } y(t) < V_{low} \\ 1 & \text{if } y(t) < V_{high.} \\ X & \text{otherwise} \end{cases} \quad (16)$$

Where V_{low} and V_{high} are real values. The common logic gates (AND, OR, NOT, XOR) gates with their truth tables including the unknown value 'X' is illustrated below.

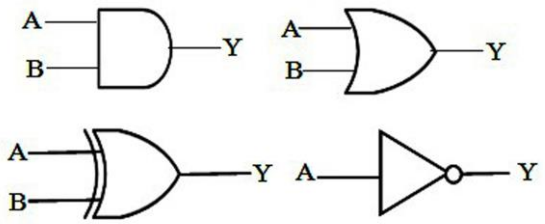


Figure 10. Basic Gates

Table 3. Truth Table of Logic Gates

Inputs		Outputs			NOT	
A	B	AND (Y)	OR (Y)	XOR (Y)	Input (A)	Output (Y)
0	0	0	0	0	0	1
0	1	0	1	1		
0	X	0	X	X		
1	0	0	1	0	1	0
1	1	1	1	1		
1	X	X	1	X		
X	0	0	X	X	X	X
X	1	X	1	X		
X	X	X	X	X		

The symbol 'X' indicates that the signal is ambiguous, *i.e.*, it may take any value. This Concept made the analysis of circuits flexible. This concept of three-valued logic is well

established and adopted for smooth analysis. This helped in decreasing the hazards of combinational circuit. Moreover, ternary extensions have widely been accepted for analysis of asynchronous circuits.

6.2. Analysis

Analysis is the process of finding the voltage across, and the current through, every component in the circuit. Analysis is just an algorithmic implementation of procedures. For those circuits to be synthesized, all of what is to be done first is nothing but analysis. The circuit designers must analyze a circuit before synthesizing it.

To analyze any circuit, we need to track the input signals right through the completion of signal propagation across the circuit to get a definite value for the period a particular input is applied. When we apply definite input values and try to follow its propagation, we fail to bring out the exact output for complex circuits. Accurate input tracking is always been troublesome, even inputs are intractable. So circuit designers face a major challenge in analyzing a circuit. This problem is resolved by the use of symbolic logic. This was given by Claude E. Shannon in 1938 [3]. It was then used for the analysis of intricate connections of relay contacts and switches. Symbolic analysis proposes formulas that define the logic values of signals. For say, symbol '+' (plus) is considered to denote series connection of two terminals and '.' (Multiply) for parallel connection. He gave general postulates which are similar to that of simple algorithms like 0+1=1 (*i.e.*, open circuit is in parallel with closed circuit) where '0' represents closed and '1' represents open circuit. He also proposed theorems like x(yz) = (xy)z, x+yz = (x+y)(x+z) etc. where x and y are hindrances or switches. This made the digital circuit interpretation easier and possibility to find the simplest circuit containing any type of connection. This has brought mathematical manipulation into the realm of digital electronics. Symbolic computation may be equated with processing all input combinations in parallel.

While analyzing electronic circuits, we may ask two types of questions: what are the values appear at the outset and when does they occur? Functional analysis and timing analysis respectively answers the same.

6.2.1. Functional Analysis: This defines a way to determine the value that would be appeared at the output of any circuit. While tracking input value, gates may be evaluated in any order. There is no sequential order of mentoring the gates in a circuit. When all gates are evaluated with the output remaining variable, then it cannot be termed as combinational. To be more precise, if the output of any circuit, after analysis give out an unknown value, say 'X', then it would said to behave sequentially [1].

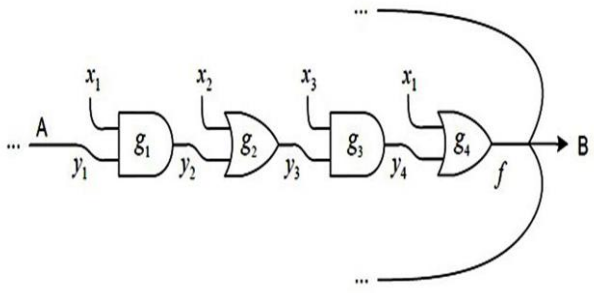


Figure 11. A Circuit Fragment

Table 4. Analysis of the Circuit Fragment

x_1	x_2	x_3	F
0	0	0	0_2
0	0	1	0_4
0	1	0	0_2
0	1	1	1_3
1	0	0	1_1
1	0	1	1_1
1	1	0	1_1
1	1	1	1_1

The circuit consists of four gates: two AND and two OR gates. Let the names of the alternating AND and OR gates are g_1, g_2, g_3, g_4 respectively.

$$\begin{aligned} g_1(x_1; y_1) &= x_1\, y_1. \\ g_2(x_2; y_2) &= x_2 + y_2 \\ g_3(x_3; y_3) &= x_3\, y_3 \\ g_4(x_1; y_4) &= x_1 + y_4. \end{aligned} \tag{17}$$

It can be said that the path between point A to B is false functionally. The gates with specific input values can be analysed as

(1) With x1 = 0, the path is blocked at gate g1.
(2) With x2 = 1, the path is blocked at gate g2.
(3) With x3 = 0, the path is blocked at gate g3.
(4) With x1 = 1, the path is blocked at gate g4.

These clearly illustrates that the path from point A to B is never sensitized.

6.2.2. Timing Analysis: This analysis puts forward an upper limit on the time of an output to occur after an input is applied. We realize the probable time when the outcome shows up. While doing so we need to work only with the knowledge of past and present input values, not future values. Evaluation of gates is done in the order that the signals proceed in a circuit. Moreover, we need the earliest time of recognizing an internal signal value to be known for evaluation of the gates. In a circuit, when we know what value appears at output, we need to know the time taken by it to occur at the output regardless of the value at the input. Assuming the Figure 9 let us consider the time taken by the corresponding gates respectively t_1, t_2, t_3, t_4.

(1) With x1 = 1, a value of 1 appears after t4 time units.
(2) With x1 = 0, and x3 = 0, a value of 0 appears after t3 + t4 time units.
(3) With x1 = 0, x3 = 1, and x2 = 1, a value of 1 appears after t2 + t3 + t4 time units.
(4)With x1 = 0, x3 = 1, and x2 = 0, a value of 0 appears after t1 + t2 + t3 + t4 time units.

If we consider $t_1=t_2=t_3=t_4=1$ i.e., we get the analysis result given in the table.

6.3. Important Parameters

For any circuit analysis, we need to go through some parameters. A parameter is an important element to consider in comprehension of an event, project or situation. Parameter has specific interpretations in mathematics and logic. Parameters, for cyclic

and acyclic circuit analysis, may include complexity, fan-in lower bound, improvement factor *etc*.

6.3.1. Complexity: We analyze circuits to evaluate the time taken, space required, or optimality etc for any logic circuit. Thus we define processes or methods to specifically determine those for a given circuit. Complexity is one way to evaluate any circuit in the fields of finding time taken, space required, performance *etc*. Time complexity and space complexity respectively determine the time taken to appear a output after an input is applied and space required implementing the circuit. In addition to that, the cost of any circuit can also be determined by complexity.

In the analysis of any circuit, we evaluate a gate whenever a new Boolean signal arrives at any of its inputs. Given a circuit with m primary inputs and n gates, each with fan-in d, there are O (dn) gate evaluations (space complexity). Moreover, timing analysis, contributes a complexity factor of O (n log2 n) (time complexity). We can perform the analysis explicitly for every assignment of input values. However, this approach is simply not tractable for most real circuits: with n variables there would be 2n input combinations to analyze separately.

6.3.2. Fan-in lower Bound: The lower bound is based on a calculation of the minimum number of gates. All existing lower bounds on circuit size are linear in the number of variables. A circuit implementing m distinct functions consists of at least m gates. An acyclic circuit implementing m distinct output functions, each depending on v input variables, consisting of gates with fan-in at most d has at least (v-1)/(d-1) + m gates. This is true for any fan-in value d ≥2 [1].

6.3.3. Improvement Factor: It is defined as the ratio of size of cyclic circuit to that of its acyclic size is bounded below by half [1]. With this improvement factor **C** our cyclic circuit becomes **C** times the size of any equivalent acyclic. Say, we have a cyclic circuit with *m* gates each gate with inputs at most *d*, that implements m functions each of which depends on all input variables *v*, then improvement factor is written as

size of cyclic circuit/size of acyclic circuit

$$=m/((v-1)/(d-1)+m-1) \tag{18}$$

Now variables *v* in a cyclic circuit is at most m(d-1), so

$$m/(m-(1/d-1)+m-1)$$
$$= m/(2m-1)\geq 1/2 \tag{19}$$

Thus if any combinational circuit with cyclic topology is implemented then that will be at most half times the size of its equivalent acyclic configuration.

6.4. Optimality

An optimal circuit refers to that which is best in every aspect. Optimality depends on a number of factors. A circuit model, obviously, puts an end to the optimality argument. Optimality varies with the varying circuit model. Secondly, if we restrict ourselves to the number of inputs applied to the gates *i.e.*, fan-in limitation, then optimality of a circuit is defined based on the inputs.

By now, we've already been familiar that some cyclic circuits are smaller in size than that of its equivalent acyclic circuit. We show that some circuits are optimal in the number of gates.

A circuit is optimal only if it satisfies the following properties.

(1) Each of the output functions depends on all its variables.

(2) The output functions are distinct.
Marc. D. Riedel [1], in his thesis, arrived with three claims for the optimality of a circuit. A cyclic circuit becomes optimal according to the following trivial claim.

(1) A circuit implementing m distinct functions consists of at least m gates.
(2) According to fan-in lower bound terminology, an acyclic circuit implementing m distinct output functions, each depending on v input variables, consisting of gates with fan-in at most d has at least {(v-1)/(d-1)+m-1}gates. This is true for any fan-in value d ≥2.

Proof.

A connected Directed Acyclic Graph (DAG) is considered. By a simple inductive argument, we can show that a connected DAG with k internal nodes each with fan-in atmost d has atmost k(d-1)+1 leaves. suppose an internal node with fan-in atmost d is added to a connected DAG. If the resulting is to be a connected DAG, the new node can replace the existing leaf. It may be attached to an existing internal node.

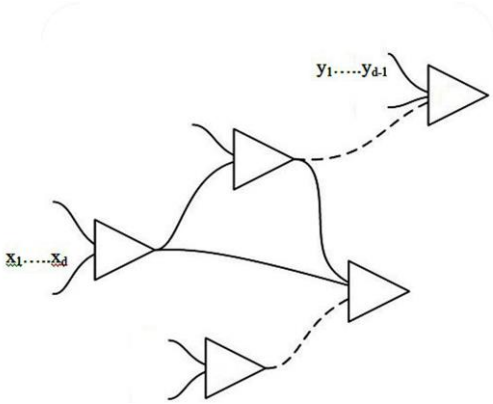


Figure 12. Adding a Node with in degree-d to Connect DAG Result in Net Gain of atmost d-1 Leaves

The first case is shown in figure with node g_1 and the latter with node g_2. In the both cases the total gain of atmost (d-1) leaves. We can conclude that connected DAG has

$$d+(k-1)(d-1) = k(d-1)+1 \quad (20)$$

leaves. Say a connected DAG has v leaves where $v \leq k(d-1)+1$ and k is bounded by

$$k \geq (v-1)/(d-1) \quad (21)$$

A circuit to be acyclic, it should have atleast one output function among the m number of output functions that depends on no other, then the out function requires atleast (v-1/d-1) gates. With distinct output functions each output function must pass through different gates. so remaining (m-1) functions require atleast (m-1) gates.

(3) The improvement factor is bounded below by ½ .

Proof.

A cyclic circuit with m gates each with fan-in d that implements m distinct output functions which depends on all input v variables, has improvement factor

$$m/((v-1)/(d-1)+m-1) \quad (2)$$

The improvement factor is minimised if the term (v-1)/(d-1) in the denominator is maximised. we know that the number of variables v in a cyclic circuit is at most m(d-1). Therefore improvement factor of such circuit can be written as

$m/(m(d-1)/(d-1))+m-1)$

$=m/(m-(1/d-1)+m-1)$

$=m/(2m-1)\geq 1/2$ (23)

7. Existing Methodologies

There are two methods being adopted till date to synthesize and analyze such circuits. They are-

(1) Branch and bound algorithm.
(2) Direct truth table method.

7.2. Branch and Bound Algorithm

This method defines to choose the output target functions so that it minimises the cost while satisfying the condition for combinationality. It was expected that the lowest cost expression for each output to be obtained with the full substitutional set (*i.e.*, all other node functions) and the highest cost expression to be obtained with the empty set. And if there are a non-trivial number of nodes, then for each node there are 2^{n-1} substitutional sets with n nodes.

Two types of branch and bound algorithms are there [1].

(a) The 'Break-Down' Approach.
(b) The 'Build-Up' Approach.

7.1.1. The 'Break-down' Approach: This type of approach needs search to be done outside the space of combinational solutions. A branch ends when it strikes a combinational solution.

The algorithm can be written as follows.

(1) The current branch must be analyzed for its combinationality. If it is combinational, it must be added to the solution list. If it is not, a set of edges must be selected to exclude based on the analysis.
(2) For each edge in the set, a new branch should be created. A node expression must be created excluding the incident node from the substitutional set. If the cost of the new branch equals or exceeds that of a solution already found, the branch should be discarded.
(3) The current branch is marked as "explored".
(4) The current branch must be set such a way that it is the lowest cost unexplored branch.
(5) The steps 1 - 4 should be repeated until the cost goal is met.

7.1.2. The 'Build-up' Approach: With this approach, the search is performed inside the space of combinational solutions. A branch ends when it hits a non-combinational solution. The search begins with an empty edge set (*i.e.*, the target functions). Edges are added as the substitution sets of nodes are augmented. As edges are included, the cost of the network remains the same or decreases.

The steps are as follows:

(1) Firstly, current branch is to be analyzed for its combinationality. If it is combinational, a set of edges must be selected to include and if it not combinational, it should be discarded.

(2) A new branch is to be created for each edge in the set. Also node expression must be created including the incident node from the substitution set.
(3) The current branch is to be marked as "explored".
(4) The current branch is to be set so it is the lowest cost unexplored branch.
(5) The above steps must be repeated until the goal is met.

7.2. Limitations of Branch and Bound Algorithm

(1) The process of branch and bound algorithm is very lengthy to define cycle in combinational circuit.
(2) The dependencies obtained in branch and bound method is not ultimate one for optimizing the logic in extreme level because further optimization is possible if we choose other dependencies.
(3) In branch and bound method Riedel used **Berkeley SIS package** which is not familiar to all.

7.3. Direct Truth Table Method

This is another method to analyze and synthesize the cyclic combinational circuits. This method was introduced by Prakash Chandra Mondal. In this method dependencies must be found out first. Dependencies are meant to be those where outputs are dependent on each other. To find the dependencies, the following rules must be followed.

(1) A table from the general truth table of the logic circuit is to be made to compare the similarity between output variables.

(2) Those outputs is to be selected which have maximum number of similarity between them.

(3) Among the outputs if we express one output in terms of another then it will be resulted in reduction of higher no of transistor counts.

(4) If we express one output in terms of another then the vice versa will not be allowed (i.e., if x is expressed in terms y then y cannot be expressed in terms of x). If we do so then the circuit will be sequential in nature.

Now we can find the output expression in terms of their dependencies by DTTM as stated below.

(1) Firstly, a table is to be made where inputs are combination of primary general inputs and secondary inputs (dependencies).

(2) The output states are to be written in the table from the general truth table where all the states of dependencies are there in combination with the primary inputs.

(3) In general truth table, only one state(either "low" or "high") for each combination of primary input will be there and the other is to be taken as "don't care" condition.

(4) After writing all the states in the cited truth table the output expressions are to be found out using any of the reduction methods we are aware of (either K- map or quine McCluskey) and hence the output functions are simplified in terms of primary and secondary inputs.

This can be explained using 2 bit magnitude comparator. The truth table is shown above.

Table 5. Truth Table of 2 Bit Magnitude Comparator

Inputs				Outputs		
B_1	B_0	A_1	A_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

The output Boolean equations after are shown below:

$$G = A_1 B_1{}' + A_0 B_0{}' B_1{}' + A_0{}' B_0{}' A_1 \quad (24)$$

$$E = B_1{}' B_0{}' A_1{}' A_0{}' + B_1{}' B_0{}' A_1 A_0 + B_1{}' B_0 A_1{}' A_0 + B_1 B_0 A_1 A_0 \quad (25)$$

$$L = A_1{}' B_1 + A_0{}' B_0 A_1{}' + A_0{}' B_0 B_1 \quad (26)$$

[Where G=A>B, E=A=B & L=A<B]L=A<B] L=A<B]

The table to determine the outputs having maximum no of similarity is given below.

Table 6. Similarity Table of 2 Bit Magnitude Comparator

Output	No. of similarity with G	No.of similarity with E	No.of similarity with L
G	-------	6	4
E	6	--------	6
L	4	6	--------

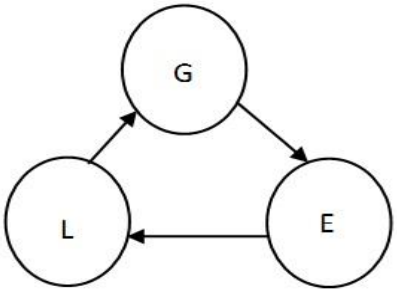
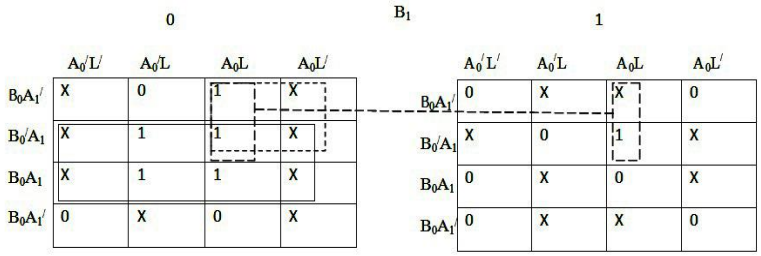


Figure 13. 2-bit Comparator Dependency Diagram

The expression of "G, E, L" in terms of their primary and secondary inputs by DTTM –The truth table of "G" in terms of its primary and secondary inputs.

Table 7. Truth Table of "G" in Terms of its Primary and Secondary Inputs

Primary and Secondary Inputs					Outputs
B_1	B_0	A_1	A_0	$L^{/}$	G
0	0	0	0	0	X
0	0	0	0	1	0
0	0	0	1	0	X
0	0	0	1	1	1
0	0	1	0	0	X
0	0	1	0	1	1
0	0	1	1	0	X
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	X
0	1	0	1	0	X
0	1	0	1	1	0
0	1	1	0	0	X
0	1	1	0	1	1
0	1	1	1	0	X
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	X
1	0	0	1	0	0
1	0	0	1	1	X
1	0	1	0	0	X
1	0	1	0	1	0
1	0	1	1	0	X
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	X
1	1	0	1	0	0
1	1	0	1	1	X
1	1	1	0	0	0
1	1	1	0	1	X
1	1	1	1	0	X
1	1	1	1	1	0



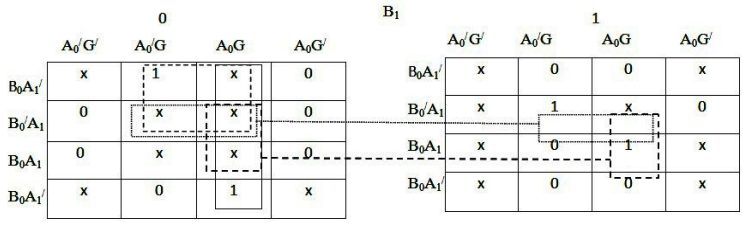
From the table above we get,

$$G = A_1 B_1^{/} + A_0 B_0^{/} L^{/} + B_1^{/} B_0^{/} A_0 \quad (27)$$

The truth table of 'E' in terms of its primary and secondary inputs.

Table 8. Truth Table of 'E' in Terms of its Primary and Secondary Inputs

Primary and secondary inputs					Output
B_1	B_0	A_1	A_0	$G^{/}$	E
0	0	0	0	0	X
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	X
0	0	1	0	0	0
0	0	1	0	1	X
0	0	1	1	0	0
0	0	1	1	1	X
0	1	0	0	0	X
0	1	0	0	1	0
0	1	0	1	0	X
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	X
0	1	1	1	0	0
0	1	1	1	1	X
1	0	0	0	0	X
1	0	0	0	1	0
1	0	0	1	0	X
1	0	0	1	1	0
1	0	1	0	0	X
1	0	1	0	1	1
1	0	1	1	0	0
1	0	1	1	1	X
1	1	0	0	0	X
1	1	0	0	1	0
1	1	0	1	0	X
1	1	0	1	1	0
1	1	1	0	0	X
1	1	1	0	1	0
1	1	1	1	0	X
1	1	1	1	1	1



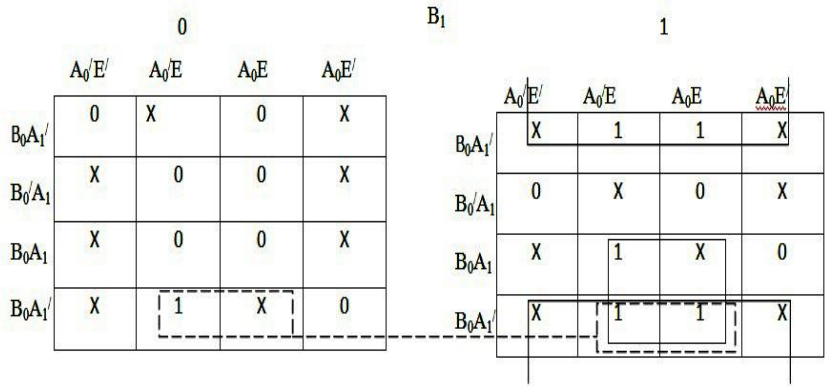
From the above table we get,

$$E=A_1 A_0 G + B_1{}^{/} B_0{}^{/} G + B_1{}^{/} A_0 G + B_0{}^{/} A_1 G \quad (28)$$

The truth table of 'L' in terms of its primary and secondary inputs

Table 9. Truth Table of 'L' in Terms of its Primary and Secondary Inputs

Primary and secondary inputs					Output
B_1	B_0	A_1	A_0	$E^/$	L
0	0	0	0	0	0
0	0	0	0	1	X
0	0	0	1	0	X
0	0	0	1	1	0
0	0	1	0	0	X
0	0	1	0	1	0
0	0	1	1	0	X
0	0	1	1	1	0
0	1	0	0	0	X
0	1	0	0	1	1
0	1	0	1	0	0
0	1	0	1	1	X
0	1	1	0	0	X
0	1	1	0	1	0
0	1	1	1	0	X
0	1	1	1	1	0
1	0	0	0	0	X
1	0	0	0	1	1
1	0	0	1	0	X
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	X
1	0	1	1	0	X
1	0	1	1	1	0
1	1	0	0	0	X
1	1	0	0	1	1
1	1	0	1	0	X
1	1	0	1	1	1
1	1	1	0	0	X
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	X



From the above K-MAP Table we get,

$$L = B_1A_1{}'+B_1B_0E+B_0A_1{}'E \quad (29)$$

So the optimized cyclic equations obtained by DTTM method are given below.

$$G = A_1B_1{}'+ A_0B_0{}'L+B_1{}'B_0{}'A_0 \quad (30)$$

$$E = A_1 A_0 G + B_1{}' B_0{}' G +B_1{}' A_0 G + B_0{}'A_1G \quad (31)$$

$$L = B_1A_1{}'+B_1B_0E+B_0A_1{}'E \quad (32)$$

Thus the equations are reduced as compared to the equations minimized in acyclic environment.

7.4. Limitation of DTTM Method

(1) Circuits should have multiple high outputs at every input combination.
For example: Decoder circuit, Demultiplexer circuit.

Truth table of 3:8 decoder circuit is given below

Table 10. Truth Table of 3:8 Decoder Circuit

Inputs			Outputs							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

The truth table shows that there is single high output at every input combination. So it is obvious, there is no similarity between each pair of outputs. According to DTTM there must have similarities between pair of outputs in order to find the dependencies which is impossible in decoder circuit. Thus DTTM fails to make such circuits into equivalent cyclic topology.
(2) Combinational circuits having outputs lesser than three cannot be made cyclic.
For example: Subtractor, Adder, and Multiplexer.

Table 11. Truth Table of Half Subtractor Circuit

Inputs		Outputs	
A	B	Difference(d)	Borrow(b)
0	0	0	0
0	0	1	0
1	1	0	0
1	1	1	1

The rule of DTTM says that if an output‘d’ is expressed by an output ‘b’ then reverse case is not possible *i.e.*, ‘b’ cannot be expressed by‘d’. In this circuit only two outputs are there. So we cannot make such circuits into cyclic.

(3) The dependencies in the DTTM method are not the ultimate one cause further optimization is possible if we take other dependencies.

For example: 7-bit Hamming code.
The truth table is given below

Table 12. Truth Table of 7-bit Hamming Code

Binary				Hamming Code (XS-3)						
X_1	X_2	X_3	X_4	A	B	C	D	E	F	G
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	1	0	0	1	1	0	0
0	0	1	0	0	1	0	0	1	0	1
0	0	1	1	1	1	0	0	1	1	0
0	1	0	0	0	0	0	1	1	1	1
0	1	0	1	1	1	1	0	0	0	0
0	1	1	0	0	0	1	1	0	0	1
0	1	1	1	1	0	1	1	0	1	0
1	0	0	0	0	1	1	0	0	1	1
1	0	0	1	0	1	1	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1
1	0	1	1	0	0	1	0	1	1	0
1	1	0	0	1	1	1	1	1	1	1
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

Table 13. Dependency Table of 7-bit Hamming Code

	A	B	C	D	E	F	G
A	------	6	6	6	6	7	5
B	6	------	7	5	7	6	6
C	6	7	------	7	5	6	6
D	6	5	7	------	7	6	6
E	6	7	5	7	------	6	6
F	7	6	6	6	6	------	7
G	5	6	6	6	6	7	------

According to the rules of DTTM method, the outputs which have maximum no of similarity between them have to be considered as dependencies. If we follow the rule we get the dependencies and diagram as shown below.

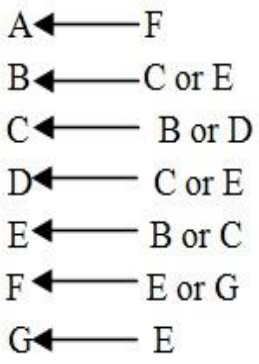
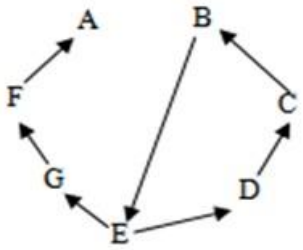


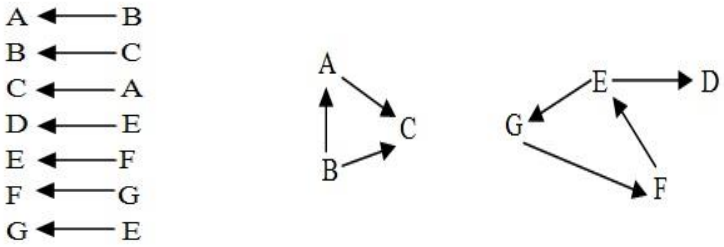
Table 14. Dependency Diagram of 7-bit Hamming code according to DTTM



Taking the above dependencies the output expressions will be

$A=X_1' X_4 + X_1 X_2 + X_2' F X_1' + X_3 F' X_1$ (33)
$B= X_3 \oplus C$ (34)
$C=X_1+X_2(X_3+D')$ (35)
$D=E X_3' + X_2 X_3$ (36)
$E=X_2' X_3+X_4 X_2'+X_2 X_3' X_4'$ (37)
$F=X_3' G + X_3 X_4$ (38)

But to get the optimum equations the dependencies should have been



The optimum output equations are given below

$A=X_1' X_4 + X_1 X_2 + X_2' X_4' B'$ (39)
$B= X_3 \oplus C$ (40)
$C= X_1 + X_2 (X_3 + A)$ (41)
$D=EX_3' + X_2 X_3$ (42)
$E=X_2'(X_4+F')+X_2 X_3' X'$ (43)
$F=X_3' G+X_3 X_4$ (44)
$G=X_4'$ (45)

A is reduced if it is made to be dependent on B. Butaccording to DTTM, it should have been dependent on F. This shows that the rule mentioned in DTTM for choosing dependencies is not correct. So it is a trial and error method.

8. Merits of Cyclic Combinational Circuits

(1) Introduction of feedback in combinational circuits maintaining its combinationality helps us to find the present output expressions without knowing past inputs.
(2) Cyclic configuration in combinational circuit reduces the size of the circuit compared to acyclic configuration of the same.
(3) Reduction in sizes of combinational circuits reduces the number of gates and hence transistors.

(4) Cyclic combinational circuits are optimum in every aspect like area and delay.

9. Demerits of Cyclic Combinational Circuits

(1) Unwanted races.
(2) Feedback in any circuit makes it less transparent. So it is hard to analyze.
(3) Timing analysis of such circuits is not very easy.
(4) Generating test routine is nightmare for test generators.

10. Conclusion and Future Scope

Cyclic circuits introduces structural feedback in the circuits which indeed increases the no. of effective input variables in the circuit & provides greater flexibility in producing minimized Boolean expression of the primary outputs. It consists of topologically multiplexed acyclic circuits. After detailed literature survey it is found that very few works has so far been reported on this topic. In this paper, we have highlighted where from the concept of cyclic circuits has come, analysis of circuit, how to design a cyclic circuit and some of the major disadvantages of it and existing methodologies, which we have to focus more. We have shown how a cyclic circuit is optimal than its acyclic equivalent.

As we have designed and shown an effective comparator circuit in cyclic environment, we should explore feedback optimizations for all types of combinational circuits, ranging from the examples that we find in text books, to the benchmark circuits referenced by the research community, to real-world circuits. If an effective decoder circuit can be made in cyclic environment we may be able to design a memory cell or ALU which will lead to the design of Processor Architecture.

Since we know that Cyclic Combinational Circuits may be potentially smaller than its acyclic versions, can we use any of the ideas developed in the circuit analysis for logic synthesis of cyclic circuits? In particular, can we incorporate this in a general logic synthesis environment, thus removing the restrictions that the circuit generated by them need to be cyclic?

We hope that this paper will help promulgate the view that a Boolean circuit is not necessarily a DAG; rather it is a directed graph that may have cycles. This distinction is a fundamental one. We have demonstrated that, it is not only feasible to design Boolean circuits with cyclic topologies, but this may well be the best way to design them.

References

[1] M. D. Riedel, “Cyclic Combinational Circuits”, California Institute of Technology, Pasadena, California, **(2004)**.
[2] P. Chandra Mondal, “Design and ASIC Implementation of Cyclic Combinational Circuits by Direct Truth Table Method For Low Power VLSI”, Submitted for Partial Fulfillment of the Requirements For the Degree of Master of Engineering In Electronics & Telecommunication Engineering, Bengal Engineering and Science University, Shibpur, **(2012)**.
[3] C. E. Shannon, “A Symbolic Analysis of Relay and Switching Circuits”, Trans. AIEE, vol. 57, **(1938)**, pp. 713-723.
[4] C. E. Shannon, “The Synthesis of Two Terminal Switching Circuits”, Bell System Technical Journal, vol. 28, **(1949)**, pp. 59-98.
[5] C. E. Shannon, “Realization of All 16 Switching Functions of Two Variables Requires 18 Contacts”, Memorandum MM53-1400-40, Bell Laboratories, **(1953)**.
[6] C. R. McCaw, “Loops in directed Combinational Switching Networks, Engineer's Thesis”, Stanford University, **(1963)**.
[7] W. H. Kautz, “The Necessity of Closed Circuit Loops in Minimal Combinational Circuits”, IEEE Trans. Comp., vol. C-19, **(1970)**, pp. 162-166.
[8] D. A. Huffman, “Combinational Circuits with Feedback”, Recent Developments in Switching Theory, A. Mukhopadhyay, **(1971)**, pp. 27-55.

[9] R. L. Rivest, "The Necessity of Feedback in Minimal Monotone Combinational Circuits", IEEE Trans. Comp., vol. C-26, no. 6, **(1977)**, pp. 606-607.
[10] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli and A. Wang, "MIS: A multiple-level logic optimization system", IEEE Trans. Computer- Aided-Design, **(1987)**, pp. 1062-1081.
[11] R. A. Short, "A Theory of Relations Between Sequential and Combinational Realizations of Switching Functions", Ph.D. Dissertation, Stanford University, **(1961)**.
[12] L. Stok, "False Loops through Resource Sharing", IBM TJ Waston Research Center, P.O Box 218, Yorktown Heights, NY, 10598.
[13] S. Malik, "Analysis of Cyclic combinational circuits", IEEE Trans. Computer- Aided-Design of Integrated Circuits and Systems, vol.1 3, no. 7, **(1994)**.
[14] T. R. Shiple, G. Berry and H. Touati, "Constructive Analysis of Cyclic Circuits", European Design and Test Conference, **(1996)**.
[15] J.-H. R. Jiang, A. Mishchenko and R. K. Brayton, "On Breakable Cyclic Definitions", Department of Electrical Engineering and Computer Sciences University of California, Berkeley.
[16] O. Neiroukh and S. A. Edwards, "Timing Analysis of Cyclic Combinational Circuits", **(2006)**.
[17] B. Mukherjee and Dr. A. Kumar Dandapat, "Design of Combinational Circuits by Cyclic Combinational Method for Low Power VLSI", International Symposium on Electronic System Design, **(2010)**.
[18] D. Ghosh, "Design of Seven Segment Decoder using Cyclic Combinational Method and its Implementation in FPGA", M.E Thesis, Bengal Engineering & Science University, Shibpur, W.B, India, **(2010)**.
[19] S. A. Edwards, "Making Cyclic Circuits Acyclic", Columbia University, Department of Computer Science New York, NY 10027, **(2003)**.
[20] M. D. Riedel and J. Bruck, Discrete applied Mathematics, vol. 160, **(2012)**, pp. 1877-1900.
[21] A. Raghunathan. P. Ashar and S. Malik, "Test Generation for Cyclic Combinational Circuits", IEEE Trans. Computer- Aided-Design of Integrated Circuits and Systems, vol. 14, no. 11, **(1995)**.
[22] J. Backes, B. Fett and M. D. Riedel, "The Analysis of Cyclic Circuits with Boolean Satisfiability", Department of Electrical and Computer Engineering University of Minnesota 200 Union St. S.E., Minneapolis, MN 55455, **(2003)**.
[23] M. D. Riedel and J. Bruck, "The Synthesis of Cyclic Combinational Circuits", California Institute of Technology, Mail Code 136–93, Pasadena, CA 91125, **(2003)**.
[24] M. D. Riedel and J. Bruck, "Timing Analysis of Cyclic Combinational Circuits", California Institute of Technology, Mail Code, Pasadena, CA 91125, pp. 136-93.
[25] V. Agarwal, N. Kankani, R. Rao, S. Bhardwaj and J. Wang, "An Efficient Combinationality Check Technique for the Synthesis of Cyclic Combinational Circuits", Electrial and Computer Engineering Department, The University of Arizona, Tucson. AZ, **(2005)**.
[26] A. Mishchenko, "An Introduction to Zero-Suppressed Binary Decision Diagrams", Berkeley Verification and Synthesis Research Center Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, **(2014)** February 5.
[27] D. Rautenbach, C. Szegedy and J. Werber, "Delay optimization of Linear Depth Boolean Circuits with Prescribed Arrival Times".
[28] E. F. Moore and C. E. Shannon, "Reliable circuits using less reliable relay".
[29] C. Y. Lee, "Representation of Switching Circuits by Binary- Decision Programs", **(1958)**.
[30] M. Mendler, "Ternary simulation: Refinement of Binary Function or abstraction Real-Time Behaviour", Springer electronics workshop and computing, **(1996)** October.
[31] C. E. Shannon, "The Systhesis of Two-Terminal Switching Circuits", Bell system Technical journal.
[32] S.-ichi Minato, "Zero-supressed BDDs for set Manipulation in combinatorial Problems", NTT LSI Laboratories, 3-1, Morinosato Wakamiya, Atsugi-shi, Kanagawa pref., 243-01, Japan.
[33] A. Jakoby and C. Schindelhauer, "On the Complexity of Worst Case and Expected Time in a Circuit", Jermany.
[34] A. Anand Kumar, "Fundamentals of Digital Circuits", PHI Learning Private Ltd, **(2009)**.
[35] M. Mano, Digital Logic and Computer Design, Prentice Hall, USA, **(1979)**.
[36] N. Upadhyay, "The Design and Analysis of Algorithms", S. K. Kataria and Sons, New Delhi, **(2011)**.

Authors

Alak Majumder, he received B.E in Electronics and Telecommunication Engineering from Tripura Institute of Technology, Agartala in 2011 and M-Tech in Microelectronics & VLSI Design from National Institute of Technology, Agartala, India in May 2013. After M-Tech he joined The ICFAI University, Tripura as a Faculty Member where he worked for 2 months. Since September'2013 he is working as an Assistant Professor in the

department of ECE at National institute of Technology, Arunachal Pradesh, India. He is an author/co-author of a several papers in International Journals and Conferences. His current research interests include Analog and Digital VLSI, High Speed Interconnects and Reversible Logic. He is a Member of IEEE, IAENG and IACSIT.

Bipasha Nath, she was born in 11th January 1993 in a small state Tripura of India. She is currently pursuing her final year B-Tech in Electronics and Telecommunication Engineering from Tripura Institute of technology Agartala. She is an author/co-author of a couple of papers in IEEE conferences. Her area of interest includes Digital Electronics & VLSI.

Durba Sarkar, she was born in 29th December 1993 in a small state Tripura of India. She is currently pursuing her final year B-Tech in Electronics and Telecommunication Engineering from Tripura Institute of technology Agartala. She is an author/co-author of a couple of papers in IEEE conferences. Her area of interest includes VLSI Design.

Moushumi Das, she was born in 11th February 1992 in a small state Tripura of India. She is currently pursuing her final year B-Tech in Electronics and Telecommunication Engineering from Tripura Institute of technology Agartala. She is an author/co-author of a couple of papers in IEEE conferences. Her research interest includes VLSI Design & Digital Logic Design.