A New
Set of Unary and Binary Operators
With
A New Algebraic System
For
Multiple-Valued Logic Systems:
THE ALGEBRA OF PRIORITY
(AOP)

By
Abu-Msameh, Ramadan K.
http://gtode.users3.50megs.com
abumsamh@emirates.net.ae

February 14, 2001

Table Of Content

Figures

Tables

Definitions

Theorems

Examples

Abstract

A New Set of Unary and Binary Operators
With A New Algebraic System
For Multiple-Valued Logic Systems:
The Algebra Of Priority
(AOP)

By
Abu-Msameh, R. K.
http://gtode.users3.50megs.com
abumsamh@emirates.net.ae

The aim of this paper is to introduce a new set of **unary** and **binary operators** with a new algebraic system that will allow the design of MVL digital circuits in a way that is **simpler and much more efficient than the traditional operators of MVL systems**. The algebra associated with these operators is called the **algebra of priority** (AOP). It is a new multi-valued multi-operational switching algebra. This newly introduced algebra was developed based on the **priority concept**. This paper (1) presents the priority concept and principle; (2) presents the development of AOP based on the priority principle; (3) presents the new binary operators of AOP which are called "**prioritors**" for binary, ternary and quaternary systems; (4) proves that the number of prioritors in a z-radix digital system is z!; (5) presents the basic intrinsic and extrinsic theorems of AOP; (6) presents the orthogonal theorem-I and II, which ***extends*** the **Post representations** of MVL functions from **two representations** (sum-of-products and product-of-sums) to **z! representations**; (7) presents the expansion theorem I and II, which ***extends*** the **Post expansions** of MVL functions from **two expansions** (sum-of-products and product-of-sums) to **z! expansions** (8) presents the **uniform image-scaling theorem** which *replaces* **DeMorgan's laws** (9) presents the **absorption theorem-III** which *replaces* **Kleene's laws.** (10) presents the **uniform degeneracy theory** which replaces the **duality theory;** (11) shows how to derive Boolean, Post algebra and Kleenean algebras from AOP and (12) presents design examples using AOP.

Keywords of AOP: algebra of priority, conservative operators, orthogonal operators, priority principle, prioritors, infimum, supremum, STAS, ITAS, degeneracy, descendants, ancestors, Image-Scaling, child-equation, parent-equation, degenerate operators, degenerate equations.

Keywords of MVL literature: multiple-valued logic, binary, ternary, quaternary, switching algebras, Boolean algebra, Post algebras, Kleenean algebra, function representations, Kleene's laws, DeMorgan's Laws, duality, Post representations, Post expansions.

A New Set of Unary and Binary Operators
With A New Algebraic System
For Multiple-Valued Logic Systems:
The Algebra Of Priority
(AOP)

1 INTRODUCTION

Background: Binary logic is an area that deals with the representation of data with two values '0' and '1'. The problem encountered with binary logic is the large number of bits that is needed to represent data. This problem is reflected at the hardware level in two well-known problems: **pinout problem** and **interconnection problem**. The **solution** to this problem was to **increase** the number of its logical values and not to limit them to two logical values. This solution **gave rise** to the development of **Multiple-valued Logic** (MVL) field, which uses **multiple logical values** to represent data. The number of these logical values is usually expected to be three or more. For example, in a four-valued system, MVL uses four values to represent data. If these values were to be numerical values, then 0,1,2, and 3 would be used. In this way, MVL solves (theoretically) the pinout problem and it simplifies circuit complexity of binary logic circuits.

Motivation: However, MVL designs digital circuits using the traditional operators MIN, MAX, MV-NOT, and complementary operators. The problem encountered in this design, is the large number of traditional operators that is needed to build up a digital circuit. This large number increases **complexity** and **interconnections** of **MVL circuits**. The more operators a MVL circuit needs, the more complex it gets and its interconnections get even more complex. A **solution** to this problem is to **increase** its **basic operators** of design and not limit them to the traditional operators. This approach will give rise to a new field called **Multiple-Operational Logic** (**MOL**), which uses **multiple-operations** from **unary** and **binary** operations to design digital circuits. **Thus, MOL is aimed at introducing into logical systems a variety of new operators that will make design more flexible than the MVL traditional operators**.

Contributions: In doing that, I developed a new set of **unary** and **binary operators** that will increase the number of **basic operators** and will make design more flexible than using the traditional operators alone. The **new unary operators** are classified into two categories: **conservative operators** (covered in §3.2) and **orthogonal operators** (covered in §3.3). The new binary operations are called **prioritors** (covered in §4.2). The number of these operators for a z-radix system is $z^2(z-1)$ for its orthogonal operators, z! for its conservative operators, and z! for its prioritors. The traditional operators are a subset of the **new operators**[1].

In 1938, **Claude Shannon** showed how the logical laws of **Boolean algebra**, founded by **George Boole** in 1849, could be used to synthesize digital circuits implemented by AND, OR, and NOT operators. Also, researchers showed how the laws of Post algebra, could be used to synthesize digital circuits implemented by

[1] *From this point and on, we will refer to the sets of* ***prioritors***, ***conservative operators***, *and* ***orthogonal operators*** *by the term "* ***AOP basic operators"*** *or* ***"the new operators"***.

MIN, MAX, MV-NOT, and complementary operators. Unfortunately, these algebras cannot be fully used with the new operators of AOP to synthesize digital circuits[2]. Boolean algebra is wonderful for the binary system but it does not work for other systems. Post algebra works for subsets of the new operators of AOP but not for all of them. Thus, Boolean and Post algebras cannot be fully used to synthesize digital circuits implemented by the various combinations of prioritors, conservative operators, and orthogonal operators[3]. With no other choice left, I developed a new algebraic system, called the **Algebra of Priority (AOP)**, that can fully serve these new operators and be used to synthesize digital circuits implemented by the various combinations out of these new operators. **Thus, AOP uses the new set of operators for circuits design and provides all the rules and procedures that lead to the design of any given digital circuit in a way that is simpler and much more efficient than the designs obtained by the traditional operators of MVL systems**. Since AOP works for large set of operators, it is described as a **multi-operational algebra** and since it works for any z-radix system, it is described as a **multi-valued algebra**. Thus, **AOP is a multi-valued, multi-operational algebra**.

In this paper, I solved two design problems (covered in §11) using the traditional operators and using the new operators. **In the first design** (covered in §11.1), I used the traditional operators, by Post algebra, and obtained the ternary multiplication operation by sum-of-products as a composition of 9 binary operators (6 MIN, 3 MAX) and 8 unary operators and by product-of-sums as a composition of 15 binary operators (9 MIN, 6 MAX) and 14 unary operators. **In the second design** (covered in §11.2), I used the traditional operators, using Post algebra, and obtained the given ternary operation by sum-of-products as a composition of 16 binary operators and 12 unary operators and by product-of-sums as a composition of 20 binary operators and 16 unary operators.

For the same two problems, I obtained different designs using the new operators of AOP. For the first problem, I designed the multiplication operation (covered in § 11.1.4) by 3 binary operators and one unary-operator. For the second example (covered in §11.2.4), I designed the circuit by 3 binary operators and one unary operator.

When we compare the designs obtained by the traditional operators and the ones obtained by the new operators of AOP, we find the following: In the first example, for the sum-of-products, AOP cuts the binary operators using its multi-operations by 66% and the unary operators by 87.5%. For the products-of-sum, AOP cuts the binary operators using its multi-operations by 80% and the unary operators by 92.85%. In the second example, for the sum-of-products, AOP cuts the binary operators using its multi-operations by 85% and the unary operators by 93.5%. For the products-of-sum, AOP cuts the binary operators using its multi-operations by 93.75% and the unary operators by 81.25%. Thus, these cuts show that the new operators can reduce circuit complexity of MVL circuits which lead to low power consumption, less propagation delay, higher speed, and less chip space.

The algebra associated with these new operators was developed based on the **priority concept** (covered in §2) from which its name was derived (The Algebra of Priority AOP)**.** Thus for a z-radix system, AOP is based on "z" logical values, z! binary

[2] *See my web site http://gtode.users3.50megs.com*
[3] *See similarities and differences between AOP and Post algebra*

operators called "**prioritors**", z! unary operators called "**conservative operators**" and $z^2(z-1)$ unary operators called "**orthogonal operators**".

It turned out that AOP is a very rich algebraic system in terms of its concepts, theorems and operators. Its results agree with the results obtained by Boolean algebra and by Post algebra and at the same time it expands the concepts and theorems of Post algebra even though it was developed totally from concepts that are completely independent of Post concepts and of Boolean concepts. For examples:

1. AOP **extends** the representations of MVL functions from **two representations** to **z! representations** (covered in §7) using its orthogonal theorems I & II. Its orthogonal theorem-II (covered in §7.3) provides a much efficient representations of MVL functions for hardware implementation than Post representations because it uses less number of binary operations.

2. AOP **extends** the expansion of MVL functions from **two expansions** to **z! expansions** (covered in §8)using its **expansions theorems I&II.**

3. AOP **extends** DeMorgan's laws by its Image-Scaling theorem (covered in § 9.2). In Boolean and Post algebras we can break the image of a binary operation by DeMorgan's laws (see Example 21) only if we use the NOT and MV-NOT operators. However, under the Image-Scaling theorem of AOP, we can break the image of a binary operation under all conservative unary operators (see Example 18).

4. AOP **extends** Kleene's laws by its **absorption theorem III** (covered in §12.1.1)**.**

5. AOP **extends** the current duality theory by its **degeneracy theory**[4] (covered in § 10). Instead of saying an operator has a **dual** (see Example 26) we say an operator has "**descendants**" or "**degenerate operators**". The number of descendants for an operator depends on the system radix and on the operator itself. For prioritors, the number of descendants depends on the system radix only and it is equal to z!. For example, it is 2 for the binary system, 6 for the ternary system and 24 for the quaternary system.

   The degeneracy theory of AOP agrees with the results of Boolean algebra since the number of descendants is always '2'**. However, it differs from Post algebra for non-binary radii.** For example, in ternary system, the MIN under AOP has six descendants rather than two under Post algebra; in the quaternary system, it has 24 descendants rather than two under Post algebra.

6. The degeneracy theory **extends** duality in binary system to be applied for other binary operators other than AND or OR. For example, the degeneracy theory **extends** the "duality" concept in Boolean algebra to cover the XOR,

---

*[4] The degeneracy theory is much like "Object-Programming theory". The concepts of "**inheritance**", "**ancestors**", "**descendants"** can be now applied to hardware as well as it is done in software. Object programming theory deals with **data** and **code**. The degeneracy theory deals with **operators** (like data) and **equations** (like code).*

and NXOR and others as well.

7. AOP **extends** the duality concept of equations by its **degeneracy theory** (covered in §10.4). Instead of saying an equation has a **dual** we say an equation has "**descendants**", "**degenerate equations**", or "**child-equations**". The number of child-equations for a **parent-equation** depends on the system radix and on the equation itself. For priority equations[5], the number of descendants depends on the system radix only and it is equal to z!. For example, it is 2 for the binary system, 6 for the ternary system and 24 for the quaternary system (see Example 34).

   The degeneracy theory of AOP agrees with the results of Boolean algebra since the number of child-equations or "descendants" is always '2'**. However, it differs from Post algebra for non-binary radii.** For example, the distribution equation in ternary system has six descendants in AOP rather than two in Post algebra.

2 AOP PRIORITY CONCEPT AND PRINCIPLE

In reality, we always face **parallel events** that occur at the same time. At some point in time, these parallel events are to be **processed sequentially** by a **processing system** that takes each event one at a time until it processes all of them. The picking process depends on the events and on how the system is programmed to handle them. To program the processing system, we use the **concepts of priority** to give each event **a distinct priority** that determines its processing order by the processing system. Based on a **priority assignment** and a **priority convention**, the processing system determines how to process these events sequentially.

For example, we imply the concept of priority in traffic control— signs: green (go), red (stop), and yellow (slow down for a stop). The vehicles arrive at the same time from different directions to a collective point (parallel events). The traffic signs (processing system) at the collective point give each direction a priority signal to handle its events. The direction that receives the green light (priority to pass by convention) allows its events (vehicles) to flow in the system structure (streets). The direction that receives the red light (priority to stop by convention) puts its events in that direction to stand by.

Another example is found in computers. A microprocessor, at some point in time, receives parallel interrupts (events), which require processing. The microprocessor, by software or hardware means, determines which interrupt should be first acknowledged for processing based on the priority of each interrupt.

2.1 Digital Systems and Digital Events

In digital systems, there are components that process data represented by parallel digital signals. The **components** are the **processing systems** and the **data signals** are the **parallel events**. In this case, we can apply the **priority concept**, as

[5] *When all the binary operators of an equation are prioritors we call it a priority equation*

seen in the traffic example and others as well, to digital systems and develop a **mathematical system** that can describe any digital system and predicts the behavior of its components. To develop such a mathematical system, we will treat **digital signals** (events) as **variables**, processing systems (processors) as **operators** (mathematical operations) and signals levels as **states** (**digits)**. The mathematical system that will be developed based on the priority concept in this paper is called "The algebra of priority (AOP)" in an analogy to "The Algebra of logic".

In summary, AOP describes a multi-valued digital system with "z" distinct **states or (logical-value)** and it refers to these states by '0', '1', ..., up to 'z-1'. The set {0, 1, …, z-1} is called the **states-set** or **(logic-set)** and each entry in the set is referred to as a **state** or **logical-value**.

2.2 Priority Concept

The logic concept is the keystone behind Boolean algebra. In a similar way, the priority concept is the keystone behind AOP. It is a universal and a natural concept that I did not create nor discovered but rather used. From the aforementioned introduction we can verbalize the priority concept in a standard statement as stated in Definition 1

Definition 1: Priority Concept
In a processing environment, the event with the **highest priority** in a group of **events** and **distinct priorities** will be **acknowledged first** by the environment processing system.

Assume we have "n" digital signals that run through an n-line data bus to a digital component in a z-radix digital system where each line is represented by an independent variable. This gives us "n" independent variables. These "n" signals are parallel events that reached the component at the same time (in theory). Assume now the component is going to process all of these events according to the priority concept by ***allowing the event with the highest priority to pass throughout its output based on a priority assignment determined by the logical-values of the signals***. This requires from us to assign a **distinct priority** for each **logical-value** in the logic-set. **Since the logic-set has "z" logical values, then there are "z" distinct priorities needed to represent these logical values**. The set of these distinct priorities is called the **priority set** and is defined to be {0, 1, 2, 3, ..., z-1} where each digit in the set is called a ***priority*** or a ***priority-value.*** The assignment of distinct priorities to the logical values of the logic-set or vice versa is called a **priority-assignment.** By convention, ***we will assume the order of priorities to be the numerical order of the priority value***s. That is, the priority with the least value represents the least priority and the priority with the highest value represents the highest priority. This statement is standardized in Definition 2. From this definition, the '0' priority represents the least priority and the "z-1" represents the highest priority.

2.3 Priority Convention

Definition 2: Priority Convention
The order of priorities is defined to be the numerical order of the *priority-values*. That is, the priority-value with the least value is the least priority and the priority-value with the highest value is the highest priority.

Example 1: On Priority Convention
In our traffic example we have three logical values (z=3): 'Red', 'Green', and 'Yellow'. Based on the priority convention of Definition 2, the priority of the 'Green' logical value is '2', the priority of the 'Yellow' logical-value is '1', and the priority of the 'Red' logical value is '0'. The highest priority value is '2', thus the 'Green' logical value has the highest priority. The least priority is '0', thus the 'Red' logical value has the least priority.

Example 2: On Priority Convention
In the quaternary system, assume the priority of each digit is defined as follows: the "1" digit has the first priority, the "3" digit has the second priority, the "0" digit has the third priority and the "2" digit has the fourth priority. From this assumption, we see that the '1' digit must have the highest priority and the '2' digit must have the least priority. According to the priority convention of Definition 2, the '1' digit has a priority of "3", the "3" digit has a priority of "2", the '0' digit has a priority of "1" and the '2' digit has a priority of "0".

At this point we need to translate the priority concept into a statement relevant to digital systems called a "priority principle" as defined in Definition 3.

Definition 3: Priority Principle
The priority principle states that "in a priority based digital system, the digital event with the highest priority in a group of digital events and distinct priorities will be acknowledged first by the processing system according to a given priority-assignment which assigns a distinct priority for each logical value in the logic-set of a z-radix digital system".

2.4 Priority Assignment

The **priority assignment** is a program by which the system will manage the processing order of its events.

Example 3: On Priority-assignment
In the quaternary digital system, we can assign a priority of 0 for the '2' logical value; 3 for the '1' logical value; 2 for the '3' logical value; 1 for the '0' logical value. Under this assignment, the logical-value "1" has the highest priority, the logical-value "3" has a less priority than the logical-value "1", the logical-value "0" has a less priority than the logical-value "3" and the logical-value "2" has a less priority than the logical-value "0".

AOP expresses the priority-assignment in a code called the "**priority s-code**" or "**priority-assignment s-code**" as defined by Definition 4.

Definition 4: Priority-Assignment Code
The **priority-assignment code** lists the 'z' distinct logical-values in a number prefixed with "zS" so that the *position of each logical-value from right* (*counting from zero*) *in that number is its priority*.

Example 4: On Priority Assignment
The s-code for the priority-assignment defined by Example 2 is written as 4S1302. The prefix is "4S" and the priority assignment is "1302". In this priority assignment: the "2" is the first digit from right with position "0" (counting from 0) the "0" is the

second digit from right with position '1', the "3" is the third digit from right with position '2', and the "1" is the fourth digit from right with position '3'. The ***position*** of each digit is the ***priority of that digit***. Thus, the "2" has a '0' priority; the "0" has a '1' priority; the "3" has a '2' priority and the "1" has a '3' priority. In a similar way, the s-code for the priority-assignment defined by Example 1 is written as 3S210.

In the s-code of the priority-assignment, the position of a digit reflects its priority relative to the other digits. Each digit has a higher priority than any other digit to its right and a less priority than any other digit to its left. Therefore, the least significant digit (the first digit from right) has the least priority of "0" and the most significant digit (the first digit from left) has the highest priority of 'z-1'.

Before we move to the next section to continue to use the priority principle to derive the binary operators of AOP, which are called **prioritors,** we have to stop at this stage and consider important and essential operations as a background to the section.

3 AOP Unary Operators and Operations

AOP unary operators are operations that operate on variables. In a z-radix digital system, there are $\mathbf{z^z}$ **unary operations**. The set of unary operators is partitioned into "m" partitions, where **"m" is the number of partitions of system radix 'Z' into positive summands**[6]. AOP uses two out of these "m" partitions in a z-radix digital system. The first partition is called the **conservative partition** and its operators are called "**conservative operators**", the second partition is called the **orthogonal partition** and its operators are called "**orthogonal operators**". In this section, we will describe only these two types of operators.

3.1 Image Operation

Unary operators are one-variable functions that map the logic-set into itself. Because we will use unary functions as unary operators, we modified the "f(x)" functional notation to the "$x^{—f}$ " operational notation as defined by Definition 5.

Definition 5: Unary Image Operator
The **unary image operator**, *denoted by "—f"*, is defined as $x^{—f}$ =f(x) where "x" is a parameter and "f" is a unary operator.

In AOP, we identify unary operators by the **unary s-code.** The s-code lists the function table of a unary operator in a string of digits starting from right to left prefixed with 'zS' where 'z' is the system radix and 'S' is a character stands for labeling the code as a system-code. Figure 5 shows the format of the unary s-code using the 4S1023 unary operator.

According to Definition 5, the unary operator must be shown to the right of the **image "—" operator**. Using f=2S01 (NOT) operator in the binary system, the following are unary image operations $0^{—f}$ =1, $1^{—f}$ =0. Under f=4S1023, $0^{—f}$ =3, $1^{—f}$ =2, $2^{—f}$ =0, $3^{—f}$ =1. Under f=4S1122, $0^{—f}$ =2, $1^{—f}$ =2, $2^{—f}$ =1, $3^{—f}$ =1. Under f=4S3023 $0^{—f}$ =3, $1^{—f}$ =2, $2^{—f}$ =0, $3^{—f}$ =3.

[6] *For example p(1)=1, p(2)=2, p(3)=3, p(4)=5, …*

3.2 Conservative Unary Operators

At some point in time, in MVL digital systems we have to take the image of a data set and then retrieve this data set at another point in time. Such cases are seen in data encryption at the hardware and software levels and in data storage devices like MVL flip-flops. Boolean algebra uses the NOT operator to convert and reconvert data. Post algebra uses the MV-NOT operator. AOP uses a more generalized set of operators to convert and reconvert data, which are called "**conservative operators**[7]" as defined by Definition 6.

Definition 6: Unary Conservative Operators
A conservative unary operator is an operator that is represented by a one-to-one function that maps the logic-set into itself.

The number of conservative operators in a z-radix digital system is equal to z! [14]p.172. Table 6 shows a list of all conservative operators in the binary, ternary and quaternary systems. The table lists conservative unary operators using the unary s-code under the "α" column.

There are two special conservative unary operators which are: Down-Del and Up-Del operators.

3.2.1 Down-Del Operator

Definition 7: Down-Del Operator '∇'
The Down-Del operator, denoted by "∇", is given by the s-code as ∇=zS(z-1)•••3210.

The **Down-Del operator** is a unary operator where the image of a variable "A" under it is always equal to the variable itself. That is, $A^{-\nabla}$ =A (this property is called the **identity property** of the Down-Del operator). For example, the Down-Del operator is ∇=2S10 in the binary system, ∇=3S210 in the ternary system and ∇=4S3210 in the quaternary system.

3.2.2 UpDel Operator

Definition 8: Up-Del Operator
The Up-Del operator, denoted by "Δ", is given by Δ=zS0123•••(z-1).

The **Up-Del operator** is a unary operator where the image of a variable "A" under it is always equal to Z-A-1. That is, $A^{-\Delta}$=Z-A-1. This operator corresponds to the MV-NOT or complement operator in Post algebra and for NOT operator in Boolean algebra. For example, the Up-Del operator is Δ=2S01 in the binary system, Δ=3S012 in the ternary system and Δ=4S0123 in the quaternary system.

3.2.3 Inverse Operator

In AOP, data retrieval is done by the use of unary operators called the **inverse unary operators** as defined by Definition 9.

[7] *AOP uses this name, because these operators preserve data and the converted data can be retrieved without any loss.*

Definition 9: Inverse Operator
If "y" is a conservative unary operator, then y^- is the inverse of "y" if and only if $(A^{—y})^{—y^-} = (A^{—y^-})^{—y} = \nabla$.

Where '∇' is the **Down-Del unary operator** (see Definition 7) and **'-'** is the **inverse operator**. If y=y-, then 'y' is called a **self-inverse operator**. Table 6 shows the inverse of all conservative operators in the binary, ternary and quaternary systems listed under the "α" and "α-" columns.

At the hardware level in AOP, conservative operators are called "**converters**" and self-inverse operators are called "**inverters**".

3.3 Orthogonal Operators

Post algebra uses the generalized complementation operators C_0, ... C_{n-1} where "n" is the order of Post algebra [4]. Boolean algebra uses the NOT complement operator. AOP uses a more generalized set of unary operators called the "**orthogonal operators**" as defined in Definition 10. The generalized complementation operators in Post algebra and the NOT operator in Boolean algebra are a subset of the orthogonal operators of AOP.

Definition 10: Unary Orthogonal Operator
The unary orthogonal operator in AOP is defined as $x^{—\Delta abc} = \{$ c if x=a and b otherwise$\}$.

The "c" is called the active-state digit, "b" is called the inactive-state digit, "a" is called the activating-digit, "x" is a parameter and "—Δ" is the **orthogonal operator** symbol. The image of "x" under a unary orthogonal operator is equal to the active-state digit when "x=a" and is equal to the inactive state digit when "x" is not equal to "a". For example: $0^{—\Delta 301}=0$, $1^{—\Delta 301}=0$, $2^{—\Delta 301}=0$, $3^{—\Delta 301}=1$. The number of unary orthogonal operators in a z-radix digital system is given by $\varphi(z)=z^2(z-1)$.Table 7 shows a list of all orthogonal operators of AOP in the quaternary, ternary and binary systems using the **Ω-code** whose format is shown in Figure 6.

3.4 Unary Operations

3.4.1 Sequential Image operation

By Definition 5 we can take the image of variables but sometimes in AOP there are situations where we have to take the image of a unary operation by another unary operation as defined in Definition 11.

Definition 11: Sequential Image Operation
Let 'f' and 'y' be two unary operators. The image of 'y' under 'f' is written as $y^{—f}$ and is obtained by taking the 'f' image of each digit in 'y'. That is $y^{—f}=(f\circ y)(x) = f(y(x))$ where 'x' is a variable .

This sequential image operation is an associative [14] p192-193 but not commutative [14] p.191.

Example 5: On Sequential Images of Unary Operators
Let f=4S3012 and y=4S0123. The image of 'y' under 'f' is $y^{—f}$ =4S0123$^{—4S3012}$ =4S2103 and the image of 'f' under 'y' is $f^{—y}$=4S3012$^{— 4S0123}$ =4S0321. Note that $f^{—y} \neq y^{—f}$ shows that this operation is not commutative.

There are two major unary operations in AOP: the **star operation** and the **costar operation**. Table 6 lists the star under the α^* column and the costar under the $\alpha\#$ column of each conservative unary operator.

3.4.2 Star Operation

Definition 12: Star Operation
The star operation of a unary operator is obtained by flipping the function table in the unary s-code so that the digit at the "ith" position becomes at the "(z-i-1)th" position and it is denoted by 'f*' and is read as ***the star of α***. The image under f* is given by $x^{— f^*}= (z-1-x)^{— f}$.

Example 6: On The Star Operation
In the binary system, for f=2S01 f*=2S10; f=2S10 f*=2S01. **In the ternary system,** for f=3S012 f*=3S210; f=3S021 f*=3S120; **In the quaternary system,** for f=4S0123, f*=4S3210; f=4S3210, f*=4S0123.

3.4.3 Costar Operation

Another operation that is related to the star operation is the **costar operation**. The "**costar**" operation generates a unary operator, say 'y', from a unary operator, say 'u', such that $u^{—y}$=u*.

Definition 13: The Costar Operation
The costar operation, denoted by '#', is defined as $f\#=f\text{-}^{—\alpha^*}$ where "f" is a conservative unary operator.

Example 7: On The Costar Operation
Let f=4S3021=QK. The f# is obtained by taking the image of its inverse by its star. Using Table 6, f-=4S3102 and f*=4S1203=Q9. Thus, the costar of α is
$f\#=f\text{-}^{— f^*}$=4S3102$^{—4S1203}$=4S1032=Q8.

4 AOP Binary Operators and Operations

4.1 Prioritors

Boolean algebra uses the AND and OR as its binary operators. Post algebra uses the MIN and MAX as its binary operators. AOP uses a more generalized set of binary operators called "***prioritors"*** as defined in Definition 14. This makes AOP a multi-operational algebra. The AND, OR, MIN, MAX are a subset of the prioritors of AOP. The number of prioritors in a z-radix digital system is equal to z! as we will prove that later in this section. The number of prioritors is 2 in the binary digital system, 6 in the ternary digital system and 24 in the quaternary digital system. Table 9 shows the number of prioritors for radices 2-31.

Before we define prioritors in AOP, we will reanalyze mathematically the priority-assignment represented by the priority s-code. The priority-assignment represents a one-to-one function that maps the ***priority set*** to the ***logic-set***. The

domain of this function is the priority set and the range of the function is the logic-set. Since x∈*priority set* and f(x)∈*logic-set,* then the function notation f(X) is read *as* ***the digit that has the 'x' priority***. For example, if f=4S3021, then f(1) is the digit that has the "1" priority which is 2. Similarly, f(0)=1; f(3)=3; f(2)=0. Since one-to-one functions are called unary **conservative operators** by AOP**,** then the $X^{—f}$ notation is read as ***the digit that has the 'x' priority*** or the image of "X" under "f". For example, if f=4S3021, then $1^{—f}$ is the digit with a priority of "1" which is 2 or the image of 1 under "f" which is 2. Similarly, $0^{—f}$ =1; $3^{—f}$ =3; $2^{—f}$ =0.

In AOP, we are interested in the priority of a given digit. Since the priority of a digit is the position of that digit in the priority s-code, then we are interested in the operator, which gives the priority of that digit as an image of the digit itself. That is, if f=4S1023 and f(A)=B, then we want the operator, say "y", that gives y(B)=A or Y(f(A))=A. Mathematically, "y" is called the inverse function of "f". In AOP, we call "y" the **conservative inverse operator** of "**f**" and it is denoted by "**f-**". For example, the inverse of f=4S1023 is f-=4S0132. Using the inverse operation, we can find the priority of any digit from the inverse of the priority-assignment. For example, if y=4S1023 then y-=4S0132 and the priority of 0 is $0^{—y^-}$ =2; of 1 is $1^{—y^-}$=3; of 2 is $2^{—y^-}$ =1; of 3 is $3^{—y^-}$=0. Table 8 lists the inverse of each priority-assignment under the "α-" column for the binary, ternary, and quaternary systems. At this point we can introduce the definition of prioritors.

AOP defines a **prioritor** as *a processing system that defines* ***distinct priorities*** *for all the* ***logical values*** *of its inputs (events) by its* ***priority-assignment*** *and* ***its output is equal to the input logical-value with the highest priority****.* Mathematically, the prioritors of AOP are defined by Definition 14 as two-event prioritors where we use the Greek alphabet "α" to refer to prioritors in general. Table 8 lists the prioritors of AOP for the binary, ternary, and quaternary systems by their priority-assignment under the 'α' column and by the function table under the 's-code' column.

Definition 14: AOP Prioritors
A prioritor, denoted by "α", is defined as $\mathbf{A\alpha B=\{A \text{ if } A^{—\alpha^-} \geq B^{—\alpha^-}\ ;\ B \text{ if } A^{—\alpha^-} \leq B^{—\alpha^-}\}}$

Definition 14 states that if "A" has a priority higher than or equal to the priority of "B" then the result is equal to "A"; if "B" has a priority higher than or equal to the priority of "A" then the result is equal to "B". In another words, the result of the "α" prioritor is equal to the variable value with the highest priority.

Table 8 lists the function table of prioritors in the binary, ternary and quaternary systems using a special coding system called the prioritors **s-code.** The prioritor s-code identifies prioritors by a string of characters. It starts with the digital system radix, 'S' suffix and the operator function table listed from right to left where after each 'z' digits there is a separating colon to simplify reading the code. Figure 7 shows the relation between the function table and the prioritor s-code using the QF=4S3210:2222:1211:0210 prioritor.

The "α" symbol in AαB represents a binary operator, while in the unary image operation, $A^{—\alpha^-}$, it represents the prioritor priority-assignment. For example, in the

quaternary system, let $\alpha=Q_7$. From Table 8, this "α" in AαB is α=4S3210:2210:1111:0010 and in $A^{\overline{\alpha}}$ is α=4S1023 with an inverse of α^{-}=4S0132.

Theorem 1: Number of Prioritors
The **number of prioritors** in a z-radix digital system is equal to z!.

Proof: By the priority principle all priorities must be distinct. Thus the assignments of priorities to the logical-values of the logic-set or vise versa is a one-to-one mapping process. For the "0" logical-value we can assign "z" priorities, for the "1" logical-value we can assign "z-1" priorities, for the "2" logical-value we can assign "z-2" priorities and for the i^{th} logical-value we can assign "z-i" priorities. Since each priority assignment to each logical-value is independent from the other assignments, then using the counting principle [14]p.3, there are z(z-1)(z-2)(z-3) ••• 2*1=z! distinct ways of assigning priorities to all the z-distinct logical-values. Hence, there are z! distinct prioritors. **Q.E.D.** See Table 9, which lists the number of prioritors for radices 2-31.

Since a prioritor represents a binary operation then it has two parameters, 'A' and 'B', written in the form of AαB where ' α' is the prioritor symbol. Using α=Q1 in Table 8, we have 0α3=0, 1α3=1, 2α3=2, 3α3=3, 0α0=0; and α=QO we have 0α3=3, 2α3=3, 0α1=1, 2α0=2, 1α1=1.

At the hardware level, prioritors are a general representation of **digital gates**. They can pass and block data flow. The signal with the least priority, which is called the prioritor **infimum signal**, is used to pass data out of the prioritor and the signal with the highest priority, which is called the prioritor **supremum signal**, is used to block the data flow.

4.2 AOP Operations

AOP Binary operations are operations that operate on prioritors. Some operate on the priority assignment and some operate on the prioritor function-table.

4.2.1 Notations, definitions and terminology

In Boolean and Post algebras, we use the '+, ∨' or '•, ∧' symbols to stand for the OR (MAX) and AND (MIN) binary operators. AOP uses the Greek alphabets as symbols to stand for prioritors in its algebraic equations. In this paper, we use the **α** symbol to stand for prioritors in general.

Each prioritor has an infimum digit and a supremum-digit, which are called the **prioritor switches**. The infimum digit (*a switch to open*) allows the data flow to pass through the output of its prioritor. The supremum digit (*a switch to close*) blocks the data flow out the prioritor. This physical process at the hardware level is expressed mathematically by AOP in the following definitions.

4.2.2 Infimum Operation

Definition 15: Infimum digit
The infimum digit of a prioritor operator, *denoted by* $\alpha^{\overline{V}}$, is a digit in the logic-set such that $\alpha^{\overline{V}}$ α A =Aα $\alpha^{\overline{V}}$ =A.

The **infimum operation** of the α operator is read as "**the infimum digit of α**" where "—V" is called the **inferiority operator**. Table 8 lists the infimum digit of all prioritors in the quaternary, ternary and binary digital systems under the '$\alpha^{—V}$' column. Using Table 8, the infimum of α=QA is $\alpha^{—V}$ =0, of α=Q5 is $\alpha^{—V}$ =2, of α=QE is $\alpha^{—V}$=1, of α=Q1 is $\alpha^{—V}$=3, of α=QO is $\alpha^{—V}$=0. In Boolean algebra, the "0" digit is the infimum digit of the OR operator (since 0+A=A). The "1" digit is the infimum digit of the AND operator (since 1•A=A).

4.2.3 Supremum-Operation

Definition 16: Supremum digit
The supremum digit of a prioritor operator, *denoted by* $\alpha^{—\Lambda}$, is a digit in the logic-set such that $\alpha^{—\Lambda}\alpha A=A\alpha$ $\alpha^{—\Lambda}=\alpha^{—\Lambda}$

The **supremum operation** of the α operator is read as "**the supremum digit of α**" where "—Λ" is called the **superiority operator**. Table 8 lists the supremum digit of all prioritors in the quaternary, ternary and binary digital systems under the '$\alpha^{—\Lambda}$' column. Using Table 8, the supremum of α=QA is $\alpha^{—\Lambda}$=1; of α=Q5 is $\alpha^{—\Lambda}$=0; of α=QE is $\alpha^{—\Lambda}$=2; of α=Q1 is $\alpha^{—\Lambda}$=0; of α=QO is $\alpha^{—\Lambda}$=3. In Boolean algebra, the "1" digit is the supremum digit of the OR operator (since 1+A=1). The "0" digit is the supremum digit of the AND operator (since 0•A=0).

4.2.4 Star Operation on Prioritors

The **star** and **costar** operations are the major unary operations in AOP that operate on the priority-assignments of prioritors. Table 8 lists the star under the α^* column and the costar under the α# column of each priority-assignment expressed by the priority s-code and by Q's, T's and B's codes for the quaternary, ternary, and binary systems. These two operations are very important operations in digital applications of AOP. They are used by STAS systems of AOP.

When we apply the star operation on a priority assignment of a prioritor, we **reverse** (transpose) the order of its priorities. Thus, ***the star of a prioritor is a prioritor.***

Example 8: On The Star Operation of Prioritors
In the binary system, for α=2S01 α^*=2S10; α=2S10 α^*=2S01. **In the ternary system,** for α=3S012 α^*=3S210; α=3S021 α^*=3S120; **In the quaternary system,** for α=4S0123, α^*=4S3210; α=4S3210, α^*=4S0123.

4.2.5 Costar Operation on Prioritors

The costar of a STAS system always corresponds to the NOT operator in Boolean, Post algebra and Kleenean algebras. It is very important operation in using prioritors to design MVL flip-flop circuits.

Example 9: On The Costar Operation of Prioritors
Let α=4S3021=QK. The α# is obtained by taking the image of its inverse by its star. Using Table 8, α-=4S3102 and α^*=4S1203=Q9. Thus, the costar of α is $\alpha\#=\alpha^{-—\alpha^*}$=4S3102$^{—4S1203}$=4S1032=Q8.

5 AOP TAS systems

In the previous section, we presented the prioritors of AOP. These prioritors form a complete family that has a complete set of algebraic theorems. These theorems enable us to use prioritors in digital circuits design.

It is very important to find out these theorems to enable us design MVL circuits using all prioritors. Assume we are going to pick up a pair of two prioritors to determine all algebraic theorems that may exist between them. There are $z!^2$ pairs of prioritors in a z-radix system. For example, Figure 8 shows all possible pairs in ternary system. Lines show pairs of different prioritors and circles show pairs of the same prioritor. For example, line "12" shows the (T1,T2) pair and line "36" shows the pair (T3,T6). Circle "11" shows the (T1,T1) pair and circle "55" shows the pair (T5,T5). So, the number of pairs is $z!^2=3!^2=36$.

AOP partitions these pairs into z! groups, where each group is called a **TAS** (**Two-Operational Algebraic System)**. Thus, we say that AOP has z! **TAS systems**. For example, there are 2 TAS systems in binary system, 6 TAS systems in ternary system and 24 TAS systems in quaternary system.

5.1 Definitions

Definition 17: TAS BASE
Every TAS system has a unique base denoted by '**π**' where **π** is a conservative unary operator.

Definition 18: TAS System
A **TAS** is defined mathematically as the set of all pairs of prioritors in the form of (**α,α!**) where ' **α!**' is called the **mate** of '**α**'.

Definition 19: Mate Operation and Operator
For any prioritor, say **α**, that belongs to a TAS there exists a **mate** denoted by **α!** (read as **mate of α**) and is defined as $\alpha!=\pi^{\overline{\alpha}}$ where "!" is called the **mate operator** and **"π"** is **TAS base**.

Another operation that is related to a TAS system is the **comate operation**. The "**comate**" operation generates a unary operator, say 'y', from a unary operator, say 'u', such that $u^{\overline{y}}=u!$

Definition 20: Comate Operation and Operator
For any prioritor that belongs to a TAS system there is a conservative unary operator called a **comate** denoted by **α?** and is defined as $\alpha?=\alpha\text{-}^{\overline{\alpha!}}$ (read as **comate of α**) and the '?' is called the comate operator.

5.2 Terminology

If the base of a TAS is equal to the Down-Del unary operator then the TAS is called an **intrinsic TAS system (ITAS)** otherwise it is called an **extrinsic TAS system.** In each z-radix system there is one intrinsic TAS system that describes the **intrinsic**

properties of prioritors. When the base of a TAS is equal to the Up-Del unary operator, we call it the **STAR TAS** (**STAS**) system**.**

When a TAS system exists in all radii it is called a **global TAS** and when it exists in some and not in the others it is called a **local TAS (LTAS).**

The ITAS and STAS systems are **global TAS systems**. Thus the number of global TAS systems is 2 and the number of local TAS systems is (z!-2) where 'z' is system radix. The binary system has no local TAS systems at all. The properties of Local TAS systems serve as a selecting factor of one radix over the other.

When the down-del operator is paired with the base of a TAS system, e.g. (∇,π), we call that pair the **ancestor pair**. All the pairs of a TAS system are called the descendants of the ancestor pair.

5.3 TAS Codes

AOP identifies TAS systems by the **s-code** and **index-code.** The s-code of a TAS is the same as its base s-code. For example, if the base is “4S1023” then the TAS s-code is ‘4S1023’. When we sort all bases in alphanumerical order, each base will have a unique index. The TAS index-code prefixes this index by 'TASc' where 'c' is a character that identifies the radix in use. For example the 4S1023 TAS has an index code of ‘TASQ7’ because the “4S1023” has an alphanumerical order of '7' and ‘Q’ represents quaternary system.

5.4 TAS Systems

Table 10 shows the **TAS systems in binary system**. The two TAS systems are TASB1 and TASB2. TASB1 is an extrinsic TAS, which generates the extrinsic properties between the B1 (AND) and B2 (OR) prioritors. TASB2 is an intrinsic TAS which generates the intrinsic **properties** of prioritors B1 (AND) and B2 (OR).

Table 11 shows the **ternary TAS systems**. There are six TAS systems, which are TAST1, TAST2, TAST3, TAST4, TAST5 and TAST6. Each TAS has its own pairs. For example, the TAST1 TAS has 6-pairs which are (T1,T6), (T2,T4), (T3,T5), (T4,T2), (T5,T3), and (T6,T1). The operator listed beside each pair is the comate unary operator. For example, the comate for the (T3,T5) pair in TAST1 is 'T4". TAST1 is the STAS system and TAST6 is the ITAS system.

In this paper, we will concentrate only on the two global TAS systems, which are the ITAS and STAS systems.

6 AOP THEOREMS

In this paper, we will only present all the theorems that are related to AOP **global TAS systems**, which are ITAS and STAS systems.

6.1 Terminology

Intrinsic theorems are theorems that describe each prioritor as single entity. Such theorems tell us the **properties** of each prioritor. **Extrinsic theorems** describe the functional behavior resulted from the interactions between two prioritors. They describe a pair of two prioritors as a single entity. For example, the **commutation property** of a prioritor is an **intrinsic property** but the **distribution property** of a prioritor is an **extrinsic property** because it describes the functional behavior of two prioritors.

Global Theorems are theorems that hold true for all TAS systems and for all radii. **Local Theorems** are theorems that do not hold true for all TAS systems and do not hold true for all radii. For example, the prioritors of the binary system have **local properties** that do not exist in other systems such as $A+A^{-}=1$. Thus, such a property is called a **local intrinsic property**. On the other hand $A\alpha\ \alpha^{-\Lambda}=\alpha^{-\Lambda}$ is a **global intrinsic property** because it holds true for all prioritors and in all radii. An **empirical theorem** is a theorem that does not have analytical proof but has an experimental proof (tested for specific radii and for specific TAS systems over all of its domain using computer software).

AOP has a large number of theorems[8], thus the process of naming each theorem is difficult. Therefore, AOP categorizes theorems into types to simplify the naming process by using an index scheme. So, each theorem in AOP has a **formula, name** and **type**. The formula spells out the action of the theorem. The name identifies the theorem and the type classifies the theorem. When it is impossible to derive a name for a property from its function or action we use the type and add an index to it. For example, we say "absorption-II theorem". The "absorption' is type and "II or 2" is the index. The basic types used so far by AOP are **static**, **absorption**, **transfer**, and **virtual**.

A theorem is said to be a **static theorem** if one of its sides is constant. For example, $A+A^{-}=1$ is a static theorem. A theorem is said to be an **absorption theorem** if there is at least one variable in one side that does not appear on the other side. For example, $A\alpha\ (A\alpha^{*}B)=A$ is an absorption theorem. A theorem is said to be a **transfer theorem** if one of its sides contains at least one orthogonal operator and the other side does not contain any orthogonal operator. For example, $A^{-\Delta\alpha-\Lambda\alpha-\Lambda C}\alpha$ ($A\ \alpha^{*}$ $B)=A\alpha B$ is a transfer theorem because the left side contains one orthogonal operator and the right side does not contain any orthogonal operator. A theorem is said to be a **virtual theorem,** if the removing of one term from one side does not change the equality of the two sides and the same variables still exist in the equation as in the original expression. For example, $(A\alpha A^{-f})\ \mu\ (A\ \alpha\ B)\ =(\ A\alpha B)$ is called a virtual theorem because the removal of the term "$(A\alpha A^{-f})\mu$" has no impact on the equation

---

[8] *Some of which I discovered and some is still undiscovered.*

results and the same variables "A" and "B" remain in the equation. If we remove "**α (Aα*B)**" form '**Aα (Aα*B)=A**' we get A=A, but the variables are not the same as in the original expression. A term that can be removed from an equation and still has no impact on the equation results is called a '**virtual term**'.

6.2 ITAS Intrinsic Theorems

The **ITAS theorems** of AOP are listed in Table 1.

Theorem 2: ITAS Theorems (19)

Table 1: ITAS Intrinsic Theorems

No	Name	Formula
1	**Del-Del Properties**	(1) $\nabla^{-\nabla}=\nabla$ (3) $\Delta^{-\Delta}=\nabla$ (5) $\nabla^{-}=\nabla$ (7) $\nabla^{*}=\Delta$ (2) $\nabla^{-\Delta}=\Delta$ (4) $\Delta^{-\nabla}=\Delta$ (6) $\Delta^{-}=\Delta$ (8) $\Delta^{*}=\nabla$
2	**Costar-Star Properties**	$\alpha\#\text{-}=\alpha\#$ $\alpha^{-\#\alpha}=\alpha^{*}$ $\alpha\#=\alpha^{*}\#$ $\alpha^{*}\# \neq \alpha\#^{*}$
3	**Sequential Inverse Theorem**[9]	$(f^{-y})^{-}=y^{-}{}^{-f-}$
4	**Sequential-Star Theorem**	$(f^{-y})^{*}=f^{*-y}$
5	**Star-Image Theorem**	$\alpha^{*}=\Delta^{-\alpha}$
6	**Comparison Theorem**	(1) If $A\leq B \Leftrightarrow A^{-\Delta}\geq B^{-\Delta}$ (2) If $A\geq B \Leftrightarrow A^{-\Delta}\leq B^{-\Delta}$
7	**Star Relative-Priority Theorem**	(1) If $A^{-\alpha-}\geq B^{-\alpha-} \Leftrightarrow A^{-\alpha*-}\leq B^{-\alpha*-}$ (2) If $A^{-\alpha-}\leq B^{-\alpha-} \Leftrightarrow A^{-\alpha*-}\geq B^{-\alpha*-}$
8	**Costar Relative-Priority Theorem**	(1) If $A^{-\alpha-}\geq B^{-\alpha-} \Leftrightarrow A^{-\alpha\#-\alpha-}\leq B^{-\alpha\#-\alpha-}$ (2) If $A^{-\alpha-}\leq B^{-\alpha-} \Leftrightarrow A^{-\alpha\#-\alpha-}\geq B^{-\alpha\#-\alpha-}$
9	**Mean Theorem**	(1) $A \nabla A^{-\Delta}\geq \frac{1}{2}(z\text{-}1)$ (2) $A \Delta A^{-\Delta}\leq \frac{1}{2}(z\text{-}1)$
10	**Generalized Mean Theorem**	(1) $(A\alpha A^{-\alpha\#})^{--\alpha}\geq \frac{1}{2}(z\text{-}1)$ (2) $(A*\alpha A^{-\alpha\#})^{--\alpha}\leq \frac{1}{2}(z\text{-}1)$
11	**Priority-Star Theorem**	$A\ \alpha^{*}B=\{A \text{ if } A^{-\alpha-}\leq B^{-\alpha-}; A \text{ if } A^{-\alpha-}\leq B^{-\alpha-}\}$
12	**Star-Theorem**	$\alpha^{**}=\alpha$
13	**Infimum-Digit Theorem**	$\alpha^{-V}=0^{-\alpha}$
14	**Supremum-Digit Theorem**	$\alpha^{-\Lambda}=(z\text{-}1)^{-\alpha}$
15	**Inferiority Theorem**	$\alpha^{-V}\ \alpha\ A = A$
16	**Superiority Theorem**	$\alpha^{-\Lambda}\ \alpha\ A = \alpha^{-\Lambda}$
17	**Idempotence Theorem**	$A\alpha A=A$
18	**Commutation Theorem**	$A\alpha B=B\alpha A$
19	**Association Theorem**	$A\alpha(B\alpha C)=(A\alpha B)\ \alpha C$

[9] *This is a well established theorem for 1-1 functions in the literature but rewritten using AOP notations*

6.3 STAS Extrinsic Theorems

Unlike Boolean and Post algebras, the number of binary operations in AOP increases very rapidly in the order of z!. Thus, AOP cannot provide a symbol for each operation. Instead, AOP uses the $\amalg$ **symbol** as a **global** symbol to stand for its **consecutive operations**. For example, $X_1\alpha X_2\alpha X_3\alpha X_4$ is written as $\prod_{i=1}^{\alpha:4} X_i$. There are four parameters associated with the $\amalg$ symbol, which are the binary operation to be repeated "α", the counting index "i", the index-starting value "i=1", and the index end-value "i=4"

This section lists all the **global extrinsic theorems** of the STAS system in Table 2. The STAS system has the form (α,α*). Its base is equal to up-del (Δ) operator.

Theorem 21: STAS Extrinsic Theorems (12)

Table 2: STAS Extrinsic Theorems

No	Name	Formula	
1	**Distribution Theorem**	$A\alpha\ (B\alpha^*C)=(A\alpha B)\ \alpha^*(A\alpha C)$	
2	**Absorption Theorem-I**	$A\alpha\ (A\alpha^*B)=A$	
3	**Absorption Theorem-II**	$(A\alpha B)\alpha(A\alpha^*C)= (A\alpha B)$	
4	**Absorption Theorem-III**	$(A\alpha A^{-\alpha\#})\alpha(B\alpha^*B^{-\alpha\#})=(A\alpha A^{-\alpha\#})$	
5	**Star-Cyclic Theorem**	(1) $\alpha^{-\Lambda}=\alpha^{*-V}$	(2) $\alpha^{-V}=\alpha^{*-\Lambda}$
6	**Costar-Cyclic Theorem**	(1) $\alpha^{-\Lambda}=\alpha^{-V-\alpha\#}$	(2) $\alpha^{-V}=\alpha^{-\Lambda-\alpha\#}$
7	**Static Theorem**	$A^{-\Delta\alpha-\Lambda\alpha-\Lambda B}\ \alpha\ A = \alpha^{-\Lambda}$	
8	**Transfer Theorem-I**	$A^{-\Delta\alpha-\Lambda\alpha-\Lambda C}\alpha\ (\ A\ \alpha^*\ B)=A\alpha B$	
9	**Uniform Image-Scaling**	$(A\alpha B)^{-f}= A^{-f}\ \alpha^{-f}\ B^{-f}$	
10	**Substitution Theorem**	$\prod_{i=0;i\neq K}^{\alpha^*:z-1} \bar{B}^{-\Delta i\bar{\alpha}^\wedge C} = \bar{B}^{\Delta KC\bar{\alpha}^\wedge}$	
11	**Inferiority Substitution**	$\prod_{j=0}^{\alpha^*:z-1} X^{-\Delta j\ \alpha^{-\Lambda}\alpha^{-V}} = \alpha^{-V}$	
12	**Superiority Substitution**	$\prod_{j=0}^{\alpha:z-1} X^{-\Delta j\alpha^{-\Lambda}\alpha^{-V}} = \alpha^{-\Lambda}$	

Table 8 lists the α and α* of the (α,α*) STAS systems in the quaternary, ternary and binary digital systems. The first pair in Table 8 in each system corresponds to the (MIN, MAX) in Post algebras which are (MIN,MAX)=(Q1,Q2), (MIN,MAX)=(T1,T6) and (AND,OR)= (B1,B2).

6.4 LTAS Theorems

AOP is different from Post Algebra. AOP has many TAS systems that cannot be covered in one paper. But here, I listed two LOCAL virtual theorems, which do not belong to the STAS system. Table 3 lists two virtual theorems for LTAS systems of binary, ternary, and quaternary systems.

Theorem 34: LTAS theorems (2)

Table 3: LTAS Theorems

No	Name	Formula
1	**Virtual Theorem-I**	$(A\alpha A^{-\alpha?})\ \alpha!\ (A\ \alpha\ B) = (A\alpha B)$
2	**Virtual Theorem-II**	$A\ \alpha(\ A^{-\alpha?}\alpha!\ B) = (A\alpha B)$

For the binary system, all the virtual theorems are satisfied and reduce to the $A+(A^{-*}B)=A+B$ or $A^*(A^- +B)=A^*B$ in Boolean algebra. For the ternary system, the virtual-I theorem exists in TAST4 and TAST5. Virtual-II theorem exists in TAST5. For the quaternary system, the virtual-I theorem exists in TASTQH, TASTQI, TASTQM, and TASTQN. Virtual-II theorem exists in TASQN.

Post algebra did not show any theorem in MVL systems that is equivalent to $A+(A^{-*}B)=A+B$ or $A^*(A^- +B)=A^*B$ in Boolean algebra. So, it is a remarkable achievement by AOP to show that there are theorems in MVL systems that are equivalent to such theorems.

The theorems of LTAS systems are very important in AOP. They enable AOP to use the power behind its set of multi-operators in representing functions.

7 AOP Orthogonal Theorems[10]

Boolean and Post algebras offer only two representations for n-variable MVL functions: sum-of-products and product-of-sums [4] p29-30 [9] p.95. Each representation requires a maximum number of $(n+1)z^{n-1}$ binary operations (MIN-MAX, AND-OR) and a maximum number of nz^n complementary functions $\{C_m(x)$ [4]$\}$. For example, for a two-variable function in the ternary system we need a maximum number of $(2+1)3^2-1=26$ binary operations {8 MINs & 18 MAXs or 8 MAXs & 18 MINs [9]p.93} and $2*3^2=18$ complementary functions.

AOP extends and enhances the representations of MVL functions. It offers z! **distinct representations** for MVL functions **instead** of **two representations**. The sum-of-products and product-of-sums representations are just two representations out of the z! distinct representations. For example, a MVL function in the quaternary system can be represented by 4!=24 representations. AOP extends the representations of MVL functions using two theorems called "**Orthogonal Theorem-I"** and "**Orthogonal Theorem-II**". Both theorems extend the number of representations of MVL functions to z!. The z! distinct representations give designers more alternate choices of representing MVL functions. It also enables designers to select the representation which starts-off with the lowest number of prioritors just before entering the minimization dilemma.

AOP enhances the notations of MVL function representations. Its notations allow the use of well-organized and compact formulas that handle hundreds of representations in high-radix systems. Before we present the orthogonal theorems of AOP, we will consider the following notation, terminology, and symbols.

---

[10] AOP has more complicated theorems to represent MVL functions other than the theorems of this section.

7.1 Notations, Terminology and Definitions

In AOP, the domain of MVL functions is treated as a vector domain with z^n vectors. The notation $f(u_n,...,u_2,u_1)$ is written as $f(\mathbf{X_S})$ where $\mathbf{X}s=(X_{sn},...,X_{S2},X_{S1})$ and 's' is the vector index in the domain. The values of the vector component X_{Sj} correspond to the values of the u_j variables where $1 \leq j \leq n$.

In AOP, some of the values in the function table are called "**trivial values**". A trivial value is the value that is equal to the supremum of the prioritor used to represent its function. A **term** is all the repeated operations carried out by the α prioritor in the representation. A term is called a **trivial term** if a trivial value appears in it**.** If there are no trivial terms in the final representation, then we call it a **start-off representation. MRV** is the **most repeated value** in the function table. **NMRV** is the **next most repeated value** in the function table.

The following symbols are used in the statistical equations associated with the orthogonal theorems I&II. (1) 'λ' is the number of α's in the representation. (2) 'δ' is the number of α*'s in the representation. (3) 'p' is the total number of prioritors in the representation. (4) 't' is the number of trivial terms in the function to be represented (5) 'τ' is the number of non-trivial terms in the equation. (6) φ is the number of orthogonal operators.

In AOP, we face situations where we have to count the number of occurrences of a digit in the range set of a MVL function. The next definition defines a counting operator that is used to express the mathematical formulas of AOP in a well-compact form.

Definition 21: Counting Operator
Let "A" be a subset of the integer numbers and let 'c' be an integer number. The expression $A^{\#c}$ is defined as the number of occurrences of the 'c' element in the A set where "#" is called the **counting operator**.

Example 10: On Counting Operator
Let f(A, B)= AβB be a function in the quaternary system where $\beta=Q_8$=4S1032= 4S3310:3210:1111:0010. The expression $\{f(A, B)\}^{\#2}$ =is the number of the occurrences of '2' in the function range set which is '1'. For simplicity in notations, we will treat the function symbol 'f' under the counting operator as its **range set** and disuse the parenthesis. Thus $f^{\#0}=5$, $f^{\#1}=7$, $f^{\#2}=1$, $f^{\#3}=3$.

7.2 Orthogonal Theorem-I

AOP uses the **orthogonal theorem-I** (Theorem 37) to represent MVL functions. The orthogonal theorem-I requires a maximum number of $(n+1)z^{n-1}$ prioritors and a maximum number of nz^n orthogonal operators. For example, a 2-variable MVL function in the ternary system can be represented by six representations with a maximum number of 26 prioritors and 18 orthogonal operators for each representation.

The sum-of-products and product-of-sums [4] p29-30 [9] p.95 representations of Post algebras are special cases of the orthogonal-I representations. Therefore, a Post representation requires a maximum number of $(n+1)z^{n-1}$ binary operators (MINs and MAXs) and a maximum number of nz^n complementary operators [5].

Theorem 37: Orthogonal Theorem-I

Theorem 7.1 Orthogonal Theorem-I: A MVL function of n-variables, say $f(x_s)$ where $X_S=(X_{Sn}, ..., X_{S2}, X_{S1})$, can be represented by using the (α, α^*) STAS systems and the unary orthogonal operators as:

$$f(\mathbf{x_s})= \coprod_{m=1}^{\alpha^*:z^n} (f(X_m)\ \alpha\ \coprod_{j=1}^{\alpha:n} X_{sj}{}^{\Delta X_{mj}\,\alpha\text{—}\Lambda\alpha\text{—}V})$$

where the number of …

1- trivial terms is $t= F^{\#\alpha\Lambda\text{—}}$
2- non-trivial terms is …….. $\tau=zn-t$
3- orthogonal operators is $\varphi=n\tau$
4- α^*'s is $\delta=\tau-1$
5- α's is …… $\lambda=n\tau-F^{\#\alpha\text{—}V}$
6- All prioritors is $P=\lambda+\delta=(n+1)\tau-1-F^{\#\alpha\text{—}V}$

The Post representations are a special case of orthogonal theorem-I when α=MIN for the sum-of-products representation and when α=MAX for the product-of-sums representation.

7.3 Orthogonal Theorem-II

AOP enhances the representations of MVL functions by the **orthogonal theorem-II** (Theorem 38). The enhancement is achieved by reducing the number of prioritors needed for the representations of MVL functions than the orthogonal-I representations by z^n. This makes the orthogonal-II representations less complex than the orthogonal-I representations. The orthogonal theorem-II requires a maximum number of nZ^{n-1} prioritors and a maximum number of nz^n orthogonal operators. For example, a 2-variable MVL function in the ternary system can be represented by six representations using the orthogonal theorem-II with a maximum number of 17 prioritors and 18 orthogonal operators (see Example 11; notes that the MIN and MAX are prioritors).

The orthogonal-II representations of AOP to MVL functions are less complex than Post representations by a maximum number of z^n binary operations. Post algebra does not have an equivalent theorem to the orthogonal theorem-II.

Theorem 38: Orthogonal Theorem-II

Theorem 7.2 Orthogonal Theorem-II: A MVL function with n-variables, say $f(x_s)$ where $X_S=(X_{Sn}, ..., X_{S2}, X_{S1})$, can be represented by using the (α, α^*) STAS systems and the unary orthogonal operators as:

$$f(\mathbf{X_s})= \coprod_{m=1}^{\alpha^*:z^n} \coprod_{j=1}^{\alpha:n} Xsj^{\Delta X_{mj}\,\alpha\text{—}\Lambda f(X_m)}$$

Where the number of ….

1- trivial terms is …….. $t= F^{\#\alpha\Lambda\text{—}}$
2- non-trivial terms is …… $\tau=zn-t$
3- orthogonal operators is $\varphi=n\tau$
4- α^*'s is ……..…$\delta=\tau-1$
5- α's is ……$\lambda=(n-1)\tau$
6- All prioritors is … ……...$P=\lambda+\delta=n\tau-1$

7.4 AOP representations of MVL functions

*How AOP represents a MVL function if given a specific **STAS** system*? In this case, we do the following steps: (1) Delete all entries in the function table which are equal to $\alpha^{-\wedge}$ (2) Select orthogonal theorem I or II for the representation. (3) Mark all entries in the function table that are equal to $\alpha^{-\vee}$ if orthogonal theorem-I was selected. (4) Transfer the function table into the selected orthogonal theorem as shown in Example 12 and Table 15.

*How AOP represents a MVL function if not given a **STAS** system?* In this case, we do the following steps to get the **lowest start-off representation** (Definition 22). (1) Find the **MRV** and **NMRV** from the function table. (2) Select a prioritor from Table 8, say α, such that $\alpha^{-\wedge}$=MRV and $\alpha^{-\vee}$=NMRV. (3) Delete all entries in the function table which are equal to the function MRV (4) Select orthogonal theorem I or II for the representation. (5) Mark all entries in the function table that are equal to the function NMRV if orthogonal theorem-I was selected. (6) Transfer the function table content into the selected orthogonal theorem. The marked values will not appear in the orthogonal-I representations because they are the infimum of α and are **irrelevant** to the orthogonal-II representation. See Example 12 and Table 15:

7.4.1 Lowest Start-Off Representation

*The **lowest start-off representation** does not mean the minimum representation. Further steps have to be carried by the theorems of AOP to get a minimum representation.*

Definition 22: Lowest Start-Off Representation
A representation is called the **lowest start-off representation** if the supremum of the prioritor used to represent the function is equal to its MRV and the infimum is equal to its NMRV. That is $\alpha^{-\wedge}$=MRV, $\alpha^{-\vee}$=NMRV.

7.4.2 Examples of MVL functions

Example 11: On MVL representations by AOP
In [9] p. 92-93 the ternary function example f(u,v) is represented by the sum-of-products in p.93. Let's represent the example using the orthogonal theorem-II of AOP. From H1, we get $\alpha=\wedge=\bullet$=MIN, $\alpha^*=\vee=+$=MAX, and the logic-set={e_0, e_1, e_2}. From H2, by the infimum theorem $\alpha^{-\vee}=e_2$, $\alpha^{*-\vee}=e0$, by the star-cyclic theorem $\alpha^{\wedge-}=\alpha^{*-\vee}=e_0$ and $\alpha^{*-\wedge}=\alpha^{-\vee}=e_2$. Thus f(u,v) can be represented in AOP by the orthogonal theorem-II as listed below. This representation has 9 MINs and 8 MAXs compared to 18 MINs and 8 MAXs by Post algebra.

$$f(u,v)=\begin{pmatrix}(u^{\Delta-e_0e_0f(e_0,e_0)}\bullet v^{-\Delta e_0e_0f(e_0,e_0)})\\ +(u^{\Delta-e_1e_0f(e_1,e_0)}\bullet v^{-\Delta e_0e_0f(e_1,e_0)})\\ +(u^{-\Delta e_2e_0f(e_2,e_0)}\bullet v^{-\Delta e_0e_0f(e_2,e_0)})\\ +(u^{-\Delta e_0e_0f(e_0,e_1)}\bullet v^{-\Delta e_1e_0f(e_0,e_1)})\\ +(u^{-\Delta e_1e_0f(e_1,e_1)}\bullet v^{-\Delta e_1e_0f(e_1,e_1)})\\ +(u^{-\Delta e_1e_0f(e_2,e_1)}\bullet v^{-\Delta e_1e_0f(e_2,e_1)})\\ +(u^{\Delta-e_0e_0f(e_0,e_2)}\bullet v^{-\Delta e_2e_0f(e_0,e_2)})\\ +(u^{-\Delta e_1e_0f(e_1,e_2)}\bullet v^{-\Delta e_2e_0f(e_1,e_2)})\\ +(u^{-\Delta e_2e_0f(e_2,e_2)}\bullet v^{-\Delta e_2e_0f(e_2,e_2)})\end{pmatrix}$$

Example 12: On Orthogonal Theorems-I&II
Let f(u,v) in Example 11 be f(u,v)=uβv where $\beta=T_4$=3S120=3S212:111:210 and the logic-set={0,1,2}. From the function table, MRV=1 and NMRV=2. Using Table 8, we select $\alpha=T_3$=3S102 since $T_3^{-\Lambda}$=MRV and T_3^{-V}=NMRV. Thus, the representation by **orthogonal theorem-I** is f(u,v)=($0\alpha u^{-\Delta 210}$ $\alpha v^{-\Delta 210}$)α*($u^{-\Delta 210}$ $\alpha v^{-\Delta 212}$) α*($u^{-\Delta 212}$ $\alpha v^{-\Delta 210}$)α*($u^{-\Delta 212}$ $\alpha v^{-\Delta 212}$) with 8 binary operations and 8 orthogonal operators and no NMRV values are appearing in the representation. The representation by **orthogonal theorem-II** is f(u,v)= ($u^{-\Delta 010}$ $\alpha v^{-\Delta 010}$)α*($u^{-\Delta 210}$ $\alpha v^{-\Delta 212}$)α*($u^{-\Delta 212}$ $\alpha v^{-\Delta 210}$)α*($u^{-\Delta 212}$ $\alpha v^{-\Delta 212}$) with 7 binary operations and 8 orthogonal operators. See Table 12. These representations can be **minimized** using the theorems of AOP. For example, using the substitution theorem we reduce the last two terms and get f(u, v)=($0\alpha u^{-\Delta 210}$ $\alpha v^{-\Delta 210}$)α*($u^{-\Delta 210}$ $\alpha v^{-\Delta 212}$)α*($u^{-\Delta 212}$ $\alpha v^{-\Delta 121}$) for the orthogonal-I representation and f(u, v)= ($u^{-\Delta 010}$ $\alpha v^{-\Delta 010}$)α*($u^{-\Delta 210}$ $\alpha v^{-\Delta 212}$)α*($u^{-\Delta 212}$ $\alpha v^{-\Delta 121}$) for the orthogonal-II representation. Further steps are needed to reach a minimum form.

8 AOP EXPANSION Theorems

AOP extends and enhances the expansions of MVL functions by two theorems called "**Expansion Theorem-I"** and "**Expansion Theorem-II**". Both theorems extend the number of expansions of MVL functions to z!. For example, a MVL function in the quaternary system can be expanded by 24 expansions. The enhancement in expansion theorem-II is achieved by reducing the number of prioritors by 'z'.

8.1 Expansion Theorem-I and II

Theorem 39: Expansion Theorem-I
A MVL function of one variable in a z-radix digital system can be expanded by using the (α, α*) STAS systems and the orthogonal operators as

$$f(x)=\prod_{m=0}^{\alpha^*:z-1}\left(f(m)\ \alpha\ X^{-\Delta m\ \alpha^{-\Lambda}\alpha^{-V}}\right)$$

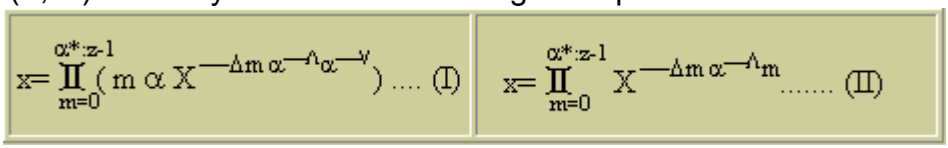
Theorem 40: Expansion Theorem-II
A MVL function of one variable in a z-radix digital system can be expanded by using the (α, α*) STAS systems and the orthogonal operators as:

$$f(x)=\prod_{m=0}^{\alpha^*:z-1} X^{-\Delta m\ \alpha^{-\Lambda f(m)}}$$

Due to the limited space, Example 14 shows only four expansions of a quaternary variable out of the 24 expansions by the orthogonal theorem-I. The first expansion is the sum-of-products by Post algebra. Also Example 15 shows four expansions of a quaternary variable out of the 24 expansions by the expansion theorem-II.

8.2 Variables Expansion

Example 13: On Variables Expansion I & II
Using the expansion theorem for f(X)=X, any variable in AOP can be expanded by using the (α,α*) STAS systems and the orthogonal operators as:



The variable expansion in Post algebra defined in axiom-3 by Epstein in (5) is a special case of the expansion in (I) when α=MIN and α*=MAX.

Example 14: On Variable Expansion-I
Using the expansion theorem-I, we can expand a quaternary variable by the following expansions:

$\alpha=Q_1=MIN$, $\alpha^*=Q_O=MAX$	$x= (1\ \alpha\ x^{-\Delta 103})\ \alpha^*(2\ \alpha\ x^{-\Delta 203})\ \alpha^*(x^{-\Delta 303}))$
$\alpha=Q_8$, $\alpha^*=Q_H$	$x= (0\ \alpha\ x^{-\Delta 012})\alpha^*\ (x^{-\Delta 212})\ \alpha^*(3\ \alpha\ x^{-\Delta 312})$
$\alpha=Q_D$, $\alpha^*=Q_L$	$x= (0\ \alpha\ x^{-\Delta 023})\alpha^*(1\ \alpha\ x^{-\Delta 123})\alpha^*\ (x^{-\Delta 323})$
$\alpha=Q_K$, $\alpha^*=Q_9$	$x= (0\ \alpha\ x^{-\Delta 031})\alpha^*(\ x^{-\Delta 131})\alpha^*(2\ \alpha\ x^{-\Delta 231})$

Example 15: On Variable Expansion-II
Using the expansion theorem-II, we can expand a quaternary variable by the following expansions:

$\alpha=Q_1$, $\alpha^*=Q_O$	$x= x^{-\Delta 101}\ \alpha^*x^{-\Delta 202}\ \alpha^*\ x^{-\Delta 303}$,	$\alpha=Q_D$, $\alpha^*=Q_L$	$x= x^{-\Delta 020}\ \alpha^*\ x^{-\Delta 121}\ \alpha^*\ x^{-\Delta 323}$
$\alpha=Q_8$, $\alpha^*=Q_H$	$x= x^{-\Delta 010}\ \alpha^*x^{-\Delta 212}\ \alpha^*\ x^{-\Delta 313}$,	$\alpha=Q_K$, $\alpha^*=Q_9$	$x= x^{-\Delta 030}\ \alpha^*\ x^{-\Delta 131}\ \alpha^*x^{-\Delta 232}$

9 AOP Image-Scaling Theorem

9.1 Binary and Priority-assignment image operations

The **Priority-assignment image operation,** denoted by the symbol '—', operates on **variables** and **on the priority-assignment of a prioritor.** For example, $\alpha^{—f}$ means that this operation is to operate on the priority-assignment of the "α" prioritor not on the prioritor itself.

Definition 23: Priority-assignment Image operation
The priority-assignment image operation, denoted by "$^{—f}$", is defined as $\alpha^{—f}$ = (priority-assignment of $\alpha)^{—f}$

The **binary image operation,** denoted by the symbol '═', operates on variables and on **the prioritor itself not on its priority-assignment**.

Definition 24: The binary image operation
The binary-image operation, denoted by $\alpha^{═f}$, is defined as $\alpha^{═f}$ =f(function Table of α).

The binary image operation and priority-assignment image operation are different and not comparable when they operate on a prioritor because the first results in a unary operator while the second results in a binary operator. The relation between the two operation is given by $(A\alpha B)^{=f}=A\alpha^{=f}\ B=(A\alpha\ B)^{-f}$

Example 16: On Prioritors Images Operations
In the quaternary system, let f=4S1302 and $\alpha=Q_J$=4S3012=4S3333:3210:3110:3000. **Using the priority-assignment image operation** α^{-f} is $_{4S3012}{}^{-4S1302}=_{4S1203}$. Note that "f" operated on the priority-assignment "4S3012" not on the prioritor "4S3333:3210:3110:3000". **Using the binary image operation** $\alpha^{=f}$ $=(4S3333:3210:3110:3000)^{=4S1302}$= 4S1111:1302:1002:1222. Note that both results are completely different and incomparable.

9.2 Uniform Image-Scaling Theorem

When we take the image of a binary operation using the NOT or MV-NOT operator in Boolean and Post algebras, we use DeMorgan's laws to break out the image operation. The DeMorgan's laws work only for the NOT and MV-NOT operators. What about if we take the image of a binary operation, say **MIN**, by using a one-to-one unary operator, say f=4S3012, other than the MV-NOT operator (see Example 18)? Post algebra does not provide the means in this case to break the image operation. AOP solves this problem by replacing DeMorgan's laws by a new theorem (Theorem 41) called the "**uniform image-scaling (UIS) theorem**" which is a special case from the General Image-Scaling Theorem.

Theorem 41: Uniform Image-Scaling (UIS) Theorem
The image of the binary operation AαB of the 'α' prioritor under a conservative unary operator 'f' is given by

$$(A\alpha B)^{-f}=A^{-f}\,\alpha^{-f}\,B^{-f}$$

9.2.1 Examples On Uniform Image-Scaling Theorem

Example 17 shows how to break up the image of the MIN binary operator under the unary operator f=4S3012. Other similar results are shown in Example 21 where "—" by default stands for the NOT (2S01) operator in the binary system and MV-NOT in MVL systems.

Example 17: On Uniform Image-Scaling Theorem (QJ,4S1302)
In the quaternary system, if we let f=4S1302 and $\alpha=Q_J$=4S3012=4S3333:3210:3110:3000, then $(A\ _{4S3012}\ B)^{-4S1302}=A^{-4S1302}\ _{4S3012}{}^{-4S1302}\ B^{-4S1302}$; $(A\ _{4S3012}\ B)^{-4S1302}=A^{-4S1302}\ _{4S1203}\ B^{-4S1302}$. According to the UIS theorem, the image on the **left side** in "$(A\ \alpha\ B)^{-f}$" is taken on the final result **not** on the priority-assignment of α, or it is taken on the function table of α as shown in "$A\ \alpha^{=f}\ B$". That is $(A\ \alpha\ B)^{=4S1302}=(A\ \alpha^{=4S1302}\ B)$ $=A\beta B$ where $\beta=(Q_J=_{4S3333:3210:3110:3000})^{=4S1302}=$ 4S1111:1302:1002:1222. On the other hand, the image operation on the **right side** is taken on the priority-assignment of α. That is $_{4S3012}{}^{-4S1302}=_{4S1203}=Q_9$. Assume A=2 and B=3, then $(2\ _{4S3012}\ 3)^{-4S1302}=2^{-4S1302}\ _{4S1203}$ $3^{-4S1302}$; $3^{-4S1302}=3\ _{4S1203}\ 1$; 1=1.

Example 18: On Uniform Image-Scaling Theorem (Q1=MIN,4S3012)
Using Table 8 and Table 14, the image of the binary operation AαB for $\alpha=Q_1$=MIN=4S0123 and f=4S3012 is $(A\ \alpha\ B)^{-f} = A^{-f}\ \alpha^{-f}\ B^{-f} = A^{-f}\beta\ B^{-f}$ where $\beta=\alpha^{-f}=\alpha^{-f}$=4S0123$^{-4SS3012}$=4S2103=$Q_F$. See Table 13, which shows the function table of this example.

Example 19: On Uniform Image-Scaling Theorem (Q1,4S0123)
In the quaternary system, let f=4S0123 and α=4S0123.

$(A\ _{4S0123}\ B)^{-4S0123} = A^{-4S0123}\ _{4S0123}\ ^{-4S0123}\ B^{-4S0123}$

$(A\ _{4S0123}\ B)^{-4S0123} = A^{-4S0123}\ _{4S3210}\ B^{-4S0123}$

$(A\ _{MIN}\ B)^{-4S0123} = A^{-4S0123}\ _{MAX}\ B^{-4S0123}$

$(A\ _{MIN}\ B)^{-} = A^{-}\ _{MAX}\ B^{-}$ using default notation where MIN=Q_1 and MAX=Q_O in Table 8.

Example 20: On Uniform Image-Scaling Theorem (T1,3S012)
In the ternary system, let f=3S012 and α=3S012.

$(A\ _{3S012}\ B)^{-3S012} = A^{-3S012}\ _{3S012}\ ^{-3S012}\ B^{-3S012}$

$(A\ _{3S012}\ B)^{-3S012} = A^{-3S012}\ _{3S210}\ B^{-3S012}$

$(A\ _{MIN}\ B)^{-3S012} = A^{-3S012}\ _{MAX}\ B^{-3S012}$

$(A\ _{MIN}\ B)^{-} = A^{-}\ _{MAX}\ B^{-}$ using default notation where MIN=T_1 and MAX=T_6 in Table 8

9.2.2 Deriving DeMorgan's laws by AOP

Example 21: On Deriving DeMorgan's Laws from UIS theorem
Since AND^{-f}=2S01^{-2S01}=2S10=OR and OR^{-f}=2S10^{-2S01}=2S01=AND when f=2S01, then we obtain $(A\ _{AND}\ B)^{-} = A^{-}\ _{OR}\ B^{-}$ and $(A\ _{OR}\ B)^{-} = A^{-}\ _{AND}\ B^{-}$, which is DeMorgan's law in Boolean algebra. Since MIN^{-f}=MAX and MAX^{-f}=MIN when f=Δ, then we obtain $(A\ _{MAX}\ B)^{-} = A^{-}\ _{MIN}\ B^{-}$ and $(A\ _{MIN}\ B)^{-} = A^{-}\ _{MAX}\ B^{-}$, which is DeMorgan's law in Post algebra.

10 AOP UNIFORM DEGENERACY

10.1 Notations and Terminology

The following AOP notation will be used in the following sections. In mathematics, we usually consider variables to be the only parameters of functions. Thus, we specify these variables in the function heading. For example, the f(x) notation means 'x' is a variable parameter and the f(x,y) means 'x' and 'y' are variable parameters. Because AOP is a multi-operational algebra, we extend the notation to specify **variables**, **operators** and **constants** as **parameters** in the function heading and at the same time use **sets notations** to specify such parameter. For example, assume we have the following Boolean function G(A,B,C)= (A+B)*(A+B)+A*C+(1+C)*(B+0). In AOP, we write this as G(X,α,C) where X={A,B,C},α={+,*}, C={1,0}. Thus, in G(x,α,c) notation (1) 'x' is the set of all variables used in the function. (2) 'α' is the set of all prioritors used in the function. (3) 'C' is the set of all constants used in the function. (4) 'G' A function whose range is determined by a set of variables 'x' and a set of prioritors 'α' and a set of constants 'c'.

Definition 25: Priority functions and Priority equations
A function or an equation is said to be a priority function or equation if and only if all of its binary operators are prioritors.

10.2 Uniform Degeneracy of Prioritors

Based on the "**duality**" concept, in Boolean and in Post algebras, we say that the dual of MIN is MAX and the dual of AND is OR and vise versa. AOP extends the duality concept into a broader scope under the concept of "**uniform degeneracy**[11]". The **uniform degeneracy of prioritors** is defined in Definition 26.

Definition 26: Uniform Degeneracy of Prioritors
The uniform degeneracy of a prioritor (**descendants**) is defined as ***the image of its priority-assignment* under a conservative unary operator**, say f, and is denoted by "α^{offf}".
Mathematically, $\alpha^{\text{offf}} \equiv \alpha^{-f}$ where α^{-f} ***is the image of the priority-assignment of*** α ***under "f" NOT the image of the function table of*** α ***under "f" and*** "**offf** " is called the uniform degeneracy operator.

Example 22: On Uniform Degeneracy of Prioritors
For example, let $\alpha=Q_7$=4S1023 and f=4S1023. By Definition 26, α^{offf} = $4S1023^{-4S1023}$= 4S2301=Q_H. Thus, we say that the uniform degeneracy of $\alpha=Q_7$ is Q_H.

Table 14 lists all the "α^{offf}" uniform degeneracy operations of all prioritors under all conservative unary operators in the quaternary, ternary and binary systems. To find the uniform degeneracy of a prioritor under a conservative unary operator using Table 14, locate the row that contains the prioritor and the column that contains the conservative unary operator, which is listed in a vertical direction. The intersection of the column and row is the prioritor number that represents the uniform degeneracy. If the number is in the quaternary system, then add the "Q" prefix; in the ternary system add the "T" prefix, in the binary system add the "B" prefix. Finally, use Table 8 to determine the function table of the prioritor found.

Example 23: On Using Uniform Degeneracy Table
The uniform degeneracy under f=4S3021 of $\alpha=Q_H$ prioritor is $Q_H^{\text{offf}}=Q_5$ (The intersection is "5" and the prefix is "Q"). The uniform degeneracy under f=3S021 of $\alpha=T_3$ prioritor is T_3offf=T_6 (The intersection is "6" and the prefix is "T"). The uniform degeneracy under f=2S01 of $\alpha=B_1$ prioritor is $B_1^{\text{offf}}=B_2$. (The intersections "2" and the prefix is "B").

Example 24: On Uniform Degeneracy of Q1 and QO
The uniform degeneracy of $\alpha=Q_1$ (MIN) under f=4S2301 is $Q_1^{\text{offf}}=Q_8$, of $\alpha=Q_1$ (MIN) under f=4S0123 is $Q_1^{\text{offf}}=Q_O$, of $\alpha=Q_O$ (MAX) under f=4S0123 is $Q_O^{\text{offf}}=Q_1$ and of $\alpha=Q_1$ (MIN) under f=4S1032 is $Q_1^{\text{offf}}=Q_H$.

[11] *AOP can extend the number of degenerate MVL equations in a z-radix system up to $z!^3$ by using its concepts of "**Non-Uniform Conservative Degeneracy**".*

Example 25: On Uniform Degeneracy of B1 (AND), B2 (OR)
In the binary system, under f=2S01, the uniform degeneracy of the AND operator (B_1) is the OR operator (B_2) and the uniform degeneracy of the OR operator (B_2) is the AND operator (B_1). That is AND^{offf}=OR and OR^{offf}=AND.

Example 26: On Duality from AOP Degeneracy
The duality theory is a special case of the uniform degeneracy theory of AOP. **The dual operation in Boolean and Post algebras is the uniform degeneracy under the up-del "Δ" conservative operator.** For example, in the quaternary system f=Δ=4S0123 and MIN=Q_1=4S0123, thus the dual of MIN is MIN^{offf}=$4S0123^{—4S0123}$= 4S3210=Q_O=MAX. In the binary system f=Δ=2S01, thus the dual of AND is AND^{offf}=$2S01^{—2S01}$=2S10=OR.

10.3 Uniform Degeneracy of Priority Functions

The duality of functions in Boolean and Post algebras is extended by AOP under the concept of "**uniform degeneracy of functions**" as defined by Definition 27.

Definition 27: Uniform Degeneracy of functions
The uniform degeneracy of a priority function, say G(x,α,c) , under a conservative unary operator, say 'f', is obtained by taking the uniform degeneracy of each prioritor in the function and by taking the image of each constant using the 'f' conservative operator where the variables of the function remain unchanged. Mathematically

$$\mathbf{G(x,\alpha,C)^{offf} = G(X,\alpha^{offf},C^{—f})}$$

The expression "$\mathbf{G(x,\alpha,C)^{offf}}$" is read as **the uniform degeneracy of the function G**. According to Definition 27, we have to take the uniform degeneracy of each prioritor and take the image of each constant and leave all variables untouched. The statement is translated symbolically as $G(X,\alpha^{offf},C^{—f})$. This means that the "^{offf} " uniform degeneracy is to operate on all the prioritors of the set 'α' and the '—f' image operator is to operate on all the constants of the set 'C'.

10.3.1 Examples on Uniform Degeneracy of Priority Functions

Example 27: On Uniform Degeneracy of functions without constants
Let *g(X,α,C)=Aα(Bα*C).* The "**offf**" uniform degeneracy of g(X,α,C) is $\mathbf{g(X,\alpha,C)^{offf} = A\alpha^{offf}(B\alpha^{*offf}C)}$.

Example 28: On Uniform Degeneracy of functions with constants
Let G(X,α,C)= (A+B)*(A+B)+A*C+(1+C)*(B+0) where x={A,B,C},α={+,*}, and C={1,0}. Using the uniform degeneracy definition, $G(A,B,C)^{offf}$=$G(X,\alpha^{offf},C^{—f})$= $G(\{A,B,C\},\{+,*\}^{offf},\{1,0\}^{—f})$= $G(\{A,B,C\},\{+^{offf},*^{offf}\},\{1^{—f},0^{—f}\})$= $(A+^{offf}B)*^{offf}(A+^{offf}B)+^{offf}A*^{offf}C+^{offf}(1^{—f}+^{offf}C)*^{offf}(B+^{offf}0^{—f})$.

Example 29: On Uniform Degeneracy of functions with constants
Let α=Q_1 (MIN) in the quaternary system and let g(**X,α,C**)=Aα(2α*B). The set of all variables in the function is X={A, B}, the set of all constants in the function is C={2}, and the set of all prioritors is α={α, α*}. The "**offf**" uniform degeneracy of g(X,α,C) is

given by $\mathbf{g(X,\alpha,C)^{offff}=A\ \alpha^{offf}\ (2^{—f}\ \alpha^{*\ offf}\ B).}$ In this example, we took the uniform degeneracy of each prioritor and took the image of the constant '2'. If we let f=4S0123, we get $\mathbf{g(X,\alpha,C)^{offf}=A\ MAX\ (1\ MIN\ B)}$ **or** $\mathbf{g(X,\alpha,C)^{offf}=A+(1\bullet B)=A\vee(1\wedge B)}$ using Boolean and Post algebra notations.

Example 30: On Uniform Degeneracy of functions with constants
Let $\alpha=Q_1$=MIN in the quaternary system and let *g(X,α,C)=(Aα(2α*B)α(Cα*3))α*(0αC).* The "**offff**" uniform degeneracy of g(X,α,C) is given by
$\mathbf{g(X,\alpha,C)^{offf}=(A\alpha^{offf}\ (2^{—f}\alpha^{*\ offf}\ B)\alpha^{offf}\ (C\alpha^{*offf}\ 3^{—f}))\ \alpha^{*offf}\ (0^{—f}\alpha^{offf}C)}$.
If f=4S1302, we obtain $\mathbf{g(X,\alpha,C)^{offf}=(AQ_E(3Q_BB)Q_E(CQ_B1))Q_B(2Q_EC).}$ Note that $\alpha^{offf}=Q_1^{offf}=4S0123^{—4S1302}=4S2031=Q_E$ and $\alpha^{*offf}=Q_O^{offf}=MAX^{offf}=4S3210^{—4S1302}=4S1302=Q_B$.

Theorem 42 shows another important theorem in AOP. This theorem gives us another way of obtaining the uniform degeneracy without using Definition 27.

Theorem 42: Uniform Degeneracy Equivalence Theorem
The uniform degeneracy of a priority function, say **G(x,α.c)**, under "offf" , where 'f' is a conservative unary operator, is equivalent to taking the image of the entire function and the inverse image of each variable in the function. Mathematically:

$$\mathbf{G(x,\alpha,c)^{offf}=G(x^{—f\ -},\alpha,c)^{—f}}$$

Example 31: On Uniform Degeneracy Equivalence Theorem
Let g(X,α,C)=Aα(Bα*C).

$G(X^{—f-},\alpha,C)^{—f}=\{A^{—f-}\alpha(B^{—f-}\alpha^{*}C^{—f-})\}^{—f}$

$=A^{—f-—f}\alpha^{offf}(B^{—f-—f}\alpha^{*\ offf}\ C^{—f-—f})$ By using the uniform image scaling theorem

$=A\ \alpha^{offf}(B\ \alpha^{*\ offf}\ C)$ By using $f^{-—f}=\nabla$

$=G(X,\alpha,C)^{offf}$ Which is the uniform degeneracy of "G"

Example 32: On Uniform Degeneracy Equivalence Theorem with constants
Let g(X,α,C)=Aα(2α*C).

$G(X^{—f-},\alpha,C)^{—f}=\{A^{—f-}\alpha(2\alpha^{*}C^{—f-})\}^{—f}$ Note that operators and constants are not affected.

$=A^{—f-—f}\alpha^{offf}(2^{—f}\alpha^{*\ offf}\ C^{—f-—f})$ By using the uniform image scaling theorem

$=A\ \alpha^{offf}(2^{—f}\alpha^{*\ offf}\ C)$ By using $f^{-—f}=\nabla$

$=G(X,\alpha,C)^{offf}$ Which is the uniform degeneracy of "G"

10.4 Uniform Degeneracy of Priority Equations

The "**duality**" theory in Boolean and in Post algebras states that out of every equation we can generate one equation, called the "dual equation", with the same variables but with different operators and constants. For example, in Boolean algebra, the dual of an equation is generated from an equation by substituting an AND for OR, an OR for AND, 0 for 1 and 1 for 0. **AOP extends the duality theory by its degeneracy theory**. AOP states that we can generate z! equations from any

given equation with the same variables but with different operators and constants. Each generated equation is called a "**child-equation**" or "**descendants**" and the original equation is called the "**parent-equation**". For example, we can generate 24 equations from a MVL equation in the quaternary system by AOP (see Example 34) but on the other hand we can generate only two equations by Post algebra. AOP in this example extends the number of equations to 24 equations $12^{(1)}$. Theorem 43 "**Equations Uniform Degeneracy Theorem**" is the mathematical statement of AOP degeneracy of equations.

Theorem 43: Equations Uniform Degeneracy Theorem
The uniform degeneracy of both sides of a **priority equation** in a z-radix digital system are equal and the **number** of uniformly degenerate equations is equal to z!. That is, if $\mathbf{G(x,c,\alpha)=H(x,c,\alpha)}$ then $\mathbf{G(x,c,\alpha)^{offf} = H(x,c,\alpha)^{offf}}$

Example 33: On Uniform Degeneracy of A+(B+1)=1
In Boolean algebra the dual of *A+(B+1)=1* is **A•(B•0)=0**. In AOP, the uniform degeneracy of *A+(B+1)=1* under the unary f=2S01 (NOT) is "$\mathbf{A+^{offf}(B+^{offf} 1^{—f})=1^{—f}}$ ". Using Table 8 and Table 14, $+^{offf}=B_2{}^{offf}=B_1$=AND=• and $1^{—f}$=0, we get **A•(B•0)=0,** which is the same result obtained by the dual operation.

Example 34: On Uniform Degeneracy of Equations with constants
In the quaternary system, let $\alpha=Q_1$ =MIN. The uniform degeneracy of the $2\alpha(B\alpha^*C)=(2\alpha B)\ \alpha^*(2\alpha C)$ MVL equation is given by $\mathbf{2^{—f}\alpha^{offf}(B\alpha^{*offf}C)=(\ 2^{—f}\alpha^{offf}\ B)\ \alpha^{*offf}(2^{—f}\alpha^{offf}C)}$**.** If f=4S2301, then we obtain $\mathbf{3_{Q_8}(B_{Q_H}C)=(\ 3_{Q_8}B)_{Q_H}(3_{Q_8}C)}$**.** Note that the Q_8 and Q_H form a STAS system that satisfies the distribution theorem. Also, note that $\alpha^{offf}=Q_1{}^{offf}$=4S0123$^{—4S2301}$=4S1032=$Q_8$ and $\alpha^{*offf}=Q_o{}^{offf}$= MAX^{offf}=4S3210$^{—4S2301}$=4S2301=Q_H.

Example 35: On Uniform Degeneracy of Distribution Theorem for z=4
Given this **parent-equation** "A∧(B∨C)=(A∧B)∨(A∧C)" in Post algebra notation, give all the 24 degenerate equations (**child-equations**) in the quaternary system where ∧=MIN and ∨=MAX. Using AOP notation, we can express **A∧(B∨C)=(A∧B)∨(A∧C) as** $A_{\alpha 1}(B\ _{\alpha o}C)=(A\ _{\alpha 1}B)_{\alpha o}(A\ _{\alpha 1}C)$ where ∧=MIN=Q_1 , ∨=MAX= Q_O. The 24 degenerate equations are listed in Table 15. The index of each α represents the prioritor number as listed in Table 9. For example, $\alpha_7=Q_7$, $\alpha_M=Q_M$ …etc. Note that the 24 uniform degeneracy operators are obtained directly from Table 14 by picking all the prioritors in a row. For example, for the z! degeneracy forms of Q_1 we pick its row in Table 14 which reads "**OIMCGA-NHK6E4-LBJ582-F9D371**" and the same for Q_O which reads "**123456-789ABC-DEFGHI-JKLMNO**". Note that the first equation is the dual of the given equation in Post algebra. This shows that the dual operation in Boolean and Post algebras is equivalent to the uniform degeneracy under the "Δ" operator. Also, note that equation No 24 is equal to the given equation, because the uniform degeneracy of any equation under the "∇" operator is equal to the equation itself.

[12] *The main reason for not adapting the same "**duality**" term is because duality implies "two" while uniform degeneracy implies two or more.*

11 Design Examples

In this section, I will present two **design examples** using the traditional operators and using AOP operators and then compare the design results.

11.1 Design of Ternary Multiplication Operation

The major operations in our lives are the basic arithmetic operations: addition, subtraction, division and multiplication. In this example, I will provide different designs for the **multiplication operation in ternary** system using AOP and Post algebras and then compare the design results. The function table for the **ternary multiplication operation** using s-code notation of AOP is 3S120:210:000.

11.1.1 Ternary Multiplier Design Using Post algebra

Example 36: Ternary Multiplier Design using Post algebra

Post algebra uses traditional operators MIN, MAX, MV-NOT, $C_0(x)$, $C_1(x)$ and $C_2(x)$. Using Post algebra notation and its representation of functions, we get the following "sum-of-products" equation (where:*=Min, +=MAX) $A*B= (1* C_1(A)* C_1(B))+ (C_1(A)*C_2(B))+ (C_2(A)* C_1(B))+ (1* C_2(A)*C_2(B))$

This **Post algebra representation** uses 9 binary operators (6 MIN, 3 MAX), and 8 unary operators (complementary functions). The corresponding circuit for this equation is shown in Figure 1. The "product-of-sums" representation uses 15 binary operators (9 MIN, 6 MAX), and 14 unary operators (complementary functions).

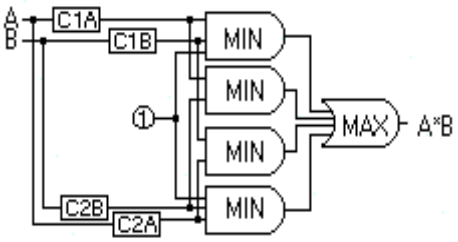


Figure 1: Ternary Multiplier By Post algebra

11.1.2 Ternary Multiplier Design Using AOP Orthogonal Theorem-I

Before we starts the design of multiplier in AOP, here are AOP operators for ternary system.

The multi-operations used by AOP in the ternary system

1. Conservative unary operators T1, T2, T3, T4, T5 and T6. (see Table 8)
2. Orthogonal operators: 3Ω001, 3Ω002, 3Ω010, 3Ω012, 3Ω020, 3Ω021, 3Ω101, 3Ω102, 3Ω110, 3Ω112, 3Ω120, 3Ω121, 3Ω201, 2Ω202, 3Ω210, 3Ω212, 3Ω220, 3Ω221
3. Prioritors: T1, T2, T3, T4, T5 and T6. (see Table 8)

In ternary system, AOP has a total of 30 operations: 24 are unary operations (out of 3^3=27 unary operations) and 6 are binary operations (out of 3^{32}=19,863 binary operations), which are its prioritors. The traditional binary operators by AOP notations are T1 (MIN), T6 (MAX) and for unary operators are T1 (MV-NOT), T6 (identity operator), 3Ω002 for $C_0(x)$, 3Ω102 for $C_1(x)$ and 3Ω202 for $C_2(x)$.

Example 37: Design using AOP Orthogonal Theorem-I

The orthogonal theorem-I is similar in format to Post representations except it is generalized to cover all the binary and unary operations of AOP. The MRV (most repeated value) in this table (3S120:210:000) is “0”. Thus we select a prioritor whose supremum-digit is equal to zero. The NMRV (next most repeated value) in this table (3S120:210:000) is ‘1’ and ‘2’. Since we have two values for NMRV, we may select a prioritor with '2' infimum-digit or with '1' infimum-digit.

The prioritor with the 1-infimum digit and 0-supremum digit is T2. Thus, we have α=T2, $\alpha^{\overline{\wedge}}$=0, $\alpha^{\overline{\vee}}$=1, α^*=T4. According to this, one of the best STAS systems to represent this function which will start-off with the lowest-representation (not minimum) is (T2,T4).
On the other hand, the prioritor with 0-supremum digit and 2-infimum digit is T1 (MIN).

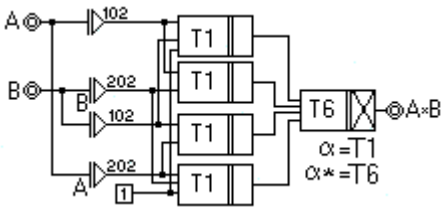


Figure 2: Ternary Multiplier By AOP Orthogonal-I

Thus, α =T1, $\alpha^{\overline{\wedge}}$=0, $\alpha^{\overline{\vee}}$=2, α^*=T6 (MAX). According to this, one of the best STAS systems to represent this function which will start-off with the lowest-representation (not minimum) is (T1,T6). Let's just use the (T1,T6) STAS system. By substituting in orthogonal theorem-I of AOP we get

$$A^*B= (1\alpha\, A^{\overline{\Delta}102}\, \alpha\, B^{\overline{\Delta}102})\, \alpha^*(\, A^{\overline{\Delta}102}\, \alpha\, B^{\overline{\Delta}202})\alpha^*(\, A^{\overline{\Delta}202}\, \alpha\, B^{\overline{\Delta}102})\alpha^*(1\alpha\, A^{\overline{\Delta}202}\, \alpha\, B^{\overline{\Delta}202})$$

Using AOP short-notation[13] we get

$$A^*B= (1\alpha\, A^{102}\, \alpha\, B^{102})\alpha^*(\, A^{102}\, \alpha\, B^{202})\alpha^*(\, A^{202}\, \alpha\, B^{102})\alpha^*(1\alpha\, A^{202}\, \alpha\, B^{202})$$

Note that this representation by AOP for this specific example is the same as of Post algebra. It uses 6 T1 (MIN) and 3 T6 (MAX). The corresponding circuit for this equation is shown in Figure 2 using AOP notations for circuits.

11.1.3 Ternary Multiplier Design Using AOP Orthogonal Theorem-II

Example 38: Design using AOP Orthogonal Theorem-II

We will use the same STAS system obtained by orthogonal theorem-I, but we substitute in orthogonal theorem-II to get the following equation:

$$A^*B= (A^{\overline{\Delta}101}\, \alpha\, B^{\overline{\Delta}101})\alpha^*(A^{\overline{\Delta}102}\alpha\, B^{\overline{\Delta}202})\alpha^*(A^{\overline{\Delta}202}\, \alpha B^{\overline{\Delta}102})\alpha^*(A^{\overline{\Delta}201}\alpha B^{\overline{\Delta}201})$$

Using AOP short-notations we get

$$A^*B= (A^{101}\, \alpha\, B^{101})\alpha^*(A^{102}\alpha\, B^{202})\alpha^*(A^{202}\, \alpha B^{102})\alpha^*(A^{201}\, \alpha\, B^{201})$$

[13] *Where "$^{\overline{\Delta}}$" operator is deleted and only left three digits.*

This representation is different from Post algebra representation and AOP orthogonal-I representation. It uses 4 T1 (MIN) and 3 T6 (MAX). The corresponding circuit for this equation is shown in Figure 3.

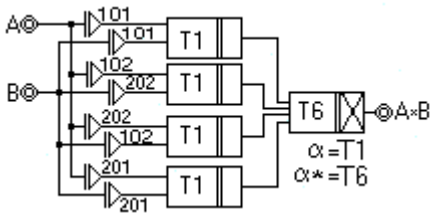


Figure 3: Ternary Multiplier By AOP Orthogonal-II

11.1.4 Ternary Multiplier Using AOP multi-operational set of basic operators

If we were to think of a different circuit for the above example, other than the one provided by Post algebra representation, then we would find it is impossible to use the MIN, MAX and MV-NOT to design such a circuit. Even the Post algebra representation used the complementary functions to get the job done.

Example 39: Using AOP multi-operational set of basic operators

Consider the circuit of Figure 4, which is drawn using AOP symbols for digital circuits. This circuit contains three prioritors labeled α, β, and μ and two conservative unary operators labeled "f" and "y" and it represents the ternary multiplication operation $A^*B= (A\alpha B^{-f})\mu\ (A^{-y}\beta B)$

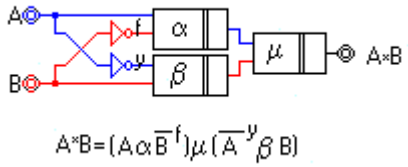


Figure 4: Ternary Multiplier By Multi-Operation set of AOP

A solution to this circuit, based on AOP multi-operators set, was carried out and gave 13 distinct circuits. A few are presented in Table-14 to the right. In comparison to the previous circuits, this circuit uses three prioritors (like saying 3 MIN) and two conservative unary operators (like saying 2 complementary functions or two orthogonal operators). By using ***AOP theorems,*** we obtain $A^*B= (A\mu B)\alpha(A\alpha B)^{-f}$ where α=T1, μ=T2, and f=T4. ***This reduces the circuit to two binary operators and one unary operator.***

Table-14

No	α	β	μ	"f"	"Y"
1	T3	T3	T5	T4	T4
2	T1	T1	T2	T4	T4
3	T1	T4	T2	T4	T3
4	T4	T1	T2	T4	T4

11.1.5 Design Comparison

Table 4 shows a summary of all designs obtained by AOP and by Post algebra. For the sum-of-products, AOP cuts the binary operators using its multi-operations by 66% and the unary operators by 87.5%. For the products-of-sum, AOP cuts the binary operators using its multi-operations by 80% and the unary operators by 92.85%.
The use of multi-operators of AOP gave us the highest reduction. So, from an engineering point of view, we would choose the design of Example 39 because it is less complex, has low power consumption, has less propagation delay, has higher speed, and uses less chip space than the other designs. From a managerial point of view, we would choose the design of Example 39 because it is more economical in terms of cost.

Table 4: Design-I Statistics and Reduction Percentages

No.	Design Methods By AOP Prioritors	Multiplications Operations			
		By Sum-of-Products		By Product-of-Sums	
		9	8	15	14
		Binary	Unary	Binary	Unary
1	Orthogonal-I	9	8	9	8
	Reduction	0%	0%	40%	42.85%
2	Orthogonal-II	7	8	7	8
	Reduction	22.22%	0%	53.33%	42.85%
3	Multi-Operational	3	1	3	1
	Reduction	66.66%	87.5%	80%	92.85%

Let's take a few of the solutions as shown in Table-14 above and discus them. In entry "1", we can use T3, and T5 prioritors and T4 conservative operator. Entry 2-1 shows another configuration. It uses T1 and T2 prioritors and T4 conservative operator. This entry shows how T2 and T4 cooperate with the traditional operator T1 to get the job done. The same can be said for the other entries.

It is impossible to design the circuit of Figure 4 using the traditional operators of Post algebra. **Thus, an engineer who relies on Post representations and has a solid faith in its traditional operators will never come up with such a circuit and will have only the circuit we obtained in design of** Example 36 **which is more complex**.

11.2 One More Design of 3S201:001:111 operation

11.2.1 Design of 3S201:001:111 Using Post algebra

Assume we are given a two-variable function with a function table given by s-code as λ=3S201:001:111 (shown to right). First we will represent the function using Post algebra and then using AOP.

AλB	0	1	2
0	1	1	1
1	1	0	0
2	1	0	2

Example 40: Design of 3S201:001:111 Using Operation using Post algebra

Assume we are given a two-variable function F(A,B) as shown. Using Post algebra notation we get the following representation (sum-of-products) $F(A,B)= (1* C_0(A)* C_0(B))+ (1* C_0(A)* C_1(B))+ (1* C_0(A)* C_2(B))+ (1* C_1(A)* C_0(B))+ (1* C_2(A)* C_0(B))+ (C_2(A)* C_2(B))$ This representation uses 16 binary operations (11 MIN and 5 MAX) and 12 unary operations.

If we use the product-of sums we get the following: $F(A,B)= (1+ J_0(A)+ J_0(B))* (1+ J_0(A)+ J_1(B))* (1+J_0(A)+J_2(B))* (1+J_1(A)+J_0(B))* (J_1(A)+ J_1(B))* (J_1(A)+ J_2(B))*(1+J_2(A)+ J_0(B))*(J_2(A)+ J_1(B))$
This product-of-sums representation uses 20 binary operations (13 MAX and 7 MIN) and 16 unary operations.

11.2.2 Design of 3S201:001:111 Using AOP Orthogonal theorem-I

For any function, Post algebra has no choices except to represent the function by MIN, MAX and complementary functions. But this is different in AOP. Since AOP has multiple operations, it does the following: it searches for the best STAS set of prioritors and then uses its orthogonal theorems to represent the function. In ternary multiplication design, AOP selected (T1,T6) and (T2,T4) as the best STAS systems and we went on and used (T1,T6) for comparison reasons with Post algebra. But

note the difference in this example, here AOP selects one STAS system based on the function to be represented.

Example 41: Design of 3S201:001:111 Using AOP Orthogonal theorem-I
The MRV (most repeated value) in the function table (3S201:001:111) of this function is "1", thus we select a prioritor whose supremum-digit is equal to one. The NMRV (next most repeated value) in this table is '0', thus we select a prioritor with an infimum-digit equal to '0'. The prioritor with 1-supremum digit and 0-infimum digit is T4. Thus α=T4, $\alpha^{-\wedge}$=1, $\alpha^{-\vee}$=0, and α^*=T2. The best STAS system to represent this function which will start-off with the lowest representation (not minimum) is (T4,T2). By substituting in AOP orthogonal theorem-I we get:

$$F(A,B)=(0\alpha A^{-\Delta 110}\alpha B^{-\Delta 110})\alpha^*(0\alpha A^{-\Delta 110}\alpha B^{-\Delta 210})\alpha(0\alpha A^{-\Delta 210}\alpha B^{-\Delta 110})\alpha(2\alpha A^{-\Delta 210}\alpha B^{-\Delta 210})$$

Since 0 is the infimum digit of α, then using the infimum-theorem, we modify the first three terms and get

$$F(A,B)= (A^{-\Delta 110}\,\alpha\, B^{-\Delta 110})\alpha^*(A^{-\Delta 110}\,\alpha\, B^{-\Delta 210})\alpha^*(A^{-\Delta 210}\,\alpha\, B^{-\Delta 110})\alpha^*(2\alpha\, A^{-\Delta\, 210}\alpha B^{-\Delta\, 210})$$

Using AOP short-notations we get

$$F(A,B)= (A^{110}\,\alpha\, B^{110})\alpha^*(A^{110}\,\alpha\, B^{210})\alpha^*(A^{210}\,\alpha\, B^{110})\alpha^*(2\alpha\, A^{210}\alpha B^{210})$$

This representation uses 5 T4s, 3 T3s, and 8 orthogonal operators. That is a total of 8 prioritors (8 binary operations) and 8 orthogonal operators. However, the Post algebra representation for the same function by sum-of-products used 16 binary operations and 12 complementary functions and by product-of-sums it used 20 binary operations and 16 unary operations. **The difference between the representations of AOP and Post algebra is 8 binary operations plus 4 unary operations for sum-of-products and 12 binary operation plus 8 unary operations for product-of-sums representation**. **This shows the expressive power of AOP over Post algebra.**

11.2.3 Design of 3S201:001:111 Using AOP Orthogonal theorem-II

Example 42: Design of 3S201:001:111 Using AOP Orthogonal theorem-II
We use the same STAS system obtained by orthogonal theorem-I, but we substitute in orthogonal theorem-II to get the following.

$$F(A,B)= (A^{-\Delta 110}\alpha\, B^{-\Delta 110})\alpha^*(A^{-\Delta 110}\,\alpha\, B^{-\Delta 210})\alpha^*(A^{-\Delta 210}\,\alpha\, B^{-\Delta 110})\alpha^*(A^{-\Delta 212}\,\alpha\, B^{-\Delta 212})$$

Using AOP short-notations

$$F(A,B)= (A^{110}\alpha\, B^{110})\alpha^*(A^{110}\,\alpha\, B^{210})\alpha^*(A^{210}\,\alpha\, B^{110})\alpha^*(A^{212}\,\alpha\, B^{212})$$

This representation uses 4 T4s and 3 T2s. That is a total of 7 binary operations and 8 orthogonal operators.

11.2.4 Design of 3S201:001:111 Using AOP multi-operators set

Example 43: Design of 3S201:001:111 Using AOP multi-operators set
Using the multi-operators set of AOP, we can design this example by 3 binary operators and 2 unary operators. A solution to this example using three prioritors and

two unary operators is $f(A,B)= (A^{-f}\alpha B)\lambda(A\ \mu B\ )^{-f}$ where α=T2, μ=T1 , λ=T3, and f=T5. **This shows how AOP reduces circuit complexity of MVL circuits. Using AOP theorems,** we can go further and reduce the equation to $f(A,B)= (A\alpha B)^{-f}$ where α=T1 and f=T5. Compare this design (1 binary operator and 1 unary operator) to Post representations (16 binary operations & 12 unary operations)!

11.2.5 Design comparison

Table 5 shows a summary of all designs obtained by AOP and by Post algebra. For the sum-of-products, AOP cuts the binary operators using its multi-operations by 85% and the unary operators by 93.5%. For the products-of-sum, AOP cuts the binary operators using its multi-operations by 93.75% and the unary operators by 81.25%.

Table 5: Design-II Statistics and Reduction Percentages

No	Design Methods By AOP Prioritors	3S201:001:111 Operation			
		By Sum-of-Products		By Products-of-Sums	
		20	16	16	12
		Binary	Unary	Binary	Unary
1	AOP Orthogonal-I	8	8	8	8
	Reduction	60%	50%	50%	33.33%
2	AOP Orthogonal-II	7	8	7	8
	Reduction	65%	50%	56.62%	33.33%
3	AOP multi-prioritors	3	1	1	1
	Reduction	85%	93.75%	93.75%	81.25

12 Deriving Other Algebras From AOP[14]

Boolean, Kleenean and **Post algebra** can be derived from AOP by (1) replacing the α prioritor by the AND or MIN and the α^* prioritor by the OR or MAX operator or vise versa; (2) replacing any orthogonal or unary operator by NOT or MV-NOT; (3) replacing any conservative unary operation by the NOT or MV-NOT operator; (4) Replacing 'z' by '2' in statistical theorems for Boolean algebra. (5) Using $MIN^{-\wedge}$=0 and $MAX^{-\wedge}$=z-1, $MIN^{-\vee}$=z-1 and $MAX^{-\vee}$=0 for Post algebra; and (5) replacing Δm0(z-1) by C_m and Δm(z-1)0 by J_m, where $J_m(X)$={0 if X=m, z-1 otherwise} $C_m(X)$={z-1 if X=m, 0 otherwise}. See next section for derivation of Kleene's laws.

12.1.1 Deriving the Kleene's Laws from the Absorption Theorem III

The Kleene's laws are: (1) $(A\bullet\bar{A})\bullet(B\vee\bar{B})= (A\bullet\bar{A})$ and (2) $(A\bullet\bar{A})\vee (B\vee\bar{B})= (B\vee\bar{B})$. The '•' operator in this law represents the MIN operator and '$\vee$' represents the MAX operator. The **DnDel** ∇ prioritor in AOP corresponds to the MAX operator and the UpDel prioritor Δ corresponds to the MIN operator.

Kleene's laws are: (1) $(A\bullet\bar{A})\bullet(B\vee\bar{B})= (A\bullet\bar{A})$ and (2) $(A\bullet\bar{A})\vee (B\vee\bar{B})= (B\vee\bar{B})$. The '•' operator in this law represents the MIN operator and '$\vee$' represents the MAX operator. The **DnDel** prioritor "**∇**" in AOP corresponds to the MAX operator and the **UpDel** prioritor "**Δ**" corresponds to the MIN operator.

[14] Visit my web site http://gtode.users3.50megs.com

Deriving the First Law

$(A\alpha A^{-\alpha\#})\alpha(B\alpha^{*}B^{-\alpha\#})=(A\alpha A^{-\alpha\#})$	Absorption theorem-III
$(A\bullet A^{-\bullet\#})\bullet(B\bullet^{*}B^{-\bullet\#})=(A\bullet A^{-\bullet\#})$	Letting $\alpha=\bullet$
$(A\bullet A^{-\bullet\#})\bullet(B\vee B^{-\bullet\#})=(A\bullet A^{-\bullet\#})$	Since $\bullet=\Delta$, then $\bullet^{*}=\Delta^{*}=\nabla=\vee$
$(A\bullet A^{-\Delta})\bullet(B\vee B^{-\Delta})=(A\bullet A^{-\Delta})$	Since $\bullet=\Delta$, then $\bullet\#=\Delta\#=\Delta\text{-}^{\Delta-^{*}}=\Delta^{-\nabla}=\Delta$
$(A\bullet \bar{A})\bullet(B\vee \bar{B})=(A\bullet\bar{A})$	Since the bar '¯' in Boolean, Post and Kleenean algebras corresponds to Δ operator in AOP. **Q.E.D.**

Deriving the Second Law

$(A\alpha A^{-\alpha\#})\alpha(B\alpha^{*}B^{-\alpha\#})=(A\alpha A^{-\alpha\#})$	Absorption theorem-III
$(A\vee A^{-\vee\#})\vee (B\vee^{*}B^{-\vee\#})=(A\vee A^{-\vee\#})$	Letting $\alpha=\vee$
$(A\vee A^{-\vee\#})\vee (B\bullet B^{-\vee\#})=(A\vee A^{-\vee\#})$	Since $\vee=\nabla$, then $\vee^{*}=\nabla^{*}=\Delta=\bullet$
$(A\vee A^{-\Delta})\vee (B\bullet B^{-\Delta})=(A\vee A^{-\Delta})$	Since $\vee=\Delta$, then $\vee\#=\nabla\#=\nabla\text{-}^{-\nabla^{*}}=\nabla^{-\Delta}=\Delta$
$(A\vee \bar{A})\vee (B\bullet \bar{B})=(A\vee \bar{A})$	Since the bar '¯' in Boolean, Post algebra and Kleenean algebras corresponds to Δ operator in AOP.
$(B\bullet \bar{B})\vee(A\vee \bar{A})=(A\vee \bar{A})$	Since $\vee$ is a commutative operator we re-order the terms on the left side
$(A\bullet \bar{A})\vee(B\vee \bar{B})=(B\vee \bar{B})$	By letting A=B and B=A which does not change the equation. This is Kleene's second law. **Q.E.D.**

13 AOP Versus Boolean and Post algebras

13.1 AOP Versus Boolean Algebra

All results obtained for the binary system by AOP are identical to that of Boolean algebra. There is no difference[15] between these two algebras at the binary system level. However, they are different for non-binary systems where Boolean algebra does not work for non-binary systems but AOP does. Also, we can derive Boolean algebra from AOP but we cannot derive AOP from Boolean algebra

There is a major difference between AOP and Boolean algebra in terms of concepts. Boolean algebra relies on "logic" concept. This concept is limited to 'true' and 'false'. It sees our world as a black and white world and it ignores the various colors of our world. On the other hand, AOP uses the priority concept, which is more global and more comprehensive concept than logic concept. The priority concept sees our world as a flux of events, which can be processed based on a priority-scheme that can be programmed in various ways to adjust to any phenomena in our world.

13.2 AOP Versus Post algebra

There are **similarities and differences** between AOP and Post algebras. The differences between AOP and Post algebra prove that AOP is not same as Post algebra and because of these differences we cannot derive AOP from Post algebra. On the other hand, we can derive Post algebra from AOP. The differences between

[15] *The general degeneracy theory expands the duality concept to include all the operators of the binary system.*

AOP and Post algebra exist on the following levels: operators, theorems, and concepts.

13.2.1 Operators Differences

The design of a MVL digital circuit in AOP is achieved by composing prioritors and unary operators into various configurations. AOP has a huge number of unary and binary operators. For example, it has 24 prioritors in the quaternary system. Post algebra is incapable of handling many of these operators. For example, the following design of the ternary multiplication[16] by AOP is composed from three prioritors labeled α, β, and μ with two conservative unary operators labeled "f" and "y" and it is written as $A*B=(A\alpha B^{-f})\mu(A^{-y}\ \beta\ B)$. There are various prioritors and conservative operators that can compose this circuit. For example, α=T1, β=T4, μ=T2, f=T4 and Y=T3. Post algebra cannot work with this equation because of the following facts: T1 and T4 do not satisfy Post algebra's axioms. Also, T1 & T2 do not satisfy Post algebra's axioms. The conservative operators T4 and T3 are not the same as MV-NOT, and thus DeMorgan's laws cannot be used. On the other hand, AOP simplifies this circuit by reducing the number of conservative operators to one instead of two: $A*B=(A\mu B)\alpha(A\alpha B)^{-f}$ where α=T1, μ=T2, and f=T4. It is obvious that Post algebra cannot handle this equation and get this reduction.

In summary, the operators domain of AOP is different from that of Post algebra and the operators domain of Post algebra is always a subset of the operators domain of AOP. So, Post algebra is a special case of AOP. This is like saying, the variables domain of Post algebra is different from that of Boolean algebra and the variables domain of Boolean algebra is always a subset of the variables domain of Post algebra. So, Boolean algebra cannot work for the entire variable domain of Post algebra. In a similar way, we say Post algebra cannot work for the entire operators domain of AOP.

13.2.2 Theorems Differences

Due to the large domain of AOP operators, it is natural to have theorems for those operators in AOP domain that do not exist in Post algebra Domain. Here are the major theorems of AOP that **do not exist** in Post algebra and **cannot be derived** from Post algebra.

- AOP Uniform Image-Scaling Theorem
- AOP Orthogonal Theorem-II
- AOP Local theorems: ex. Virtual theorems
- AOP Uniform Degeneracy theorems
- AOP non-uniform Degeneracy & Image-Scaling Theorem[17]

AOP Uniform Image-Scaling (UIS) Theorem is one of the most powerful theorems in AOP and it replaces DeMorgan's Laws. The UIS theorem simply breaks the image of binary operations under any conservative unary operator into three components as $(A\alpha B)^{-f}=A^{-f}\ \alpha^{-f}\ B^{-f}$ where "f" is a conservative operator, 'α' is a prioritor and the

[16] *See design section*

[17] *AOP non-uniform Image-Scaling Theorem and AOP non-uniform Degeneracy Theorem are completely beyond the scope of this paper.*

image operation acting on 'α' is applied to its priority-assignment. This break up is useful when this binary operation exist in equations that need to be simplified or minimized. Post algebra does not have a similar theorem that operates for any image operation done by a one-to-one function. If we use functional notation in mathematics, which is not a good practice, we would write this theorem as f(AαB)=f(A) f(α) f(B).

AOP Orthogonal Theorem-II does not exist in Post algebra and cannot be derived from Post algebra because its orthogonal operators do not satisfy Post algebra conditions of "complementation functions". Also, this theorem has better representations to MVL functions than the Post representations. This theorem also offers z! representations for any function.

AOP Local theorems help us determine which radix is best and can serve us better than other radices. Post algebra does not have local theorems for systems. Take the virtual theorem: Aα($A^{\neg f}$ β B)=AαB, where α=T1, β=T2, and f=T4 in ternary system. This theorem does no exist in Post algebra. Even though this theorem exists in Boolean algebra, in the form A+($\bar{A}$* B)=A+B or A*($\bar{A}$ +B)=A*B, Post algebra could not inherit this theorem but AOP did.

AOP Uniform Degeneracy theorem is also one of the most powerful theorems in AOP and it replaces the concepts of duality in Post algebra. Duality means that a MVL equation can have two different forms with different operators and constants but with the same variables. AOP uniform degeneracy theorem states that every MVL equation can have Z! different forms with different operators and constants but with the same variables. Unlike AOP, Post algebra follows the same steps of Boolean algebra and sates that a MVL equation or function has two different forms.

13.2.3 Concepts Differences

Due to the limitations imposed by logic concept on our world, researchers tried to find middle states between these two states ('true' and 'false') and developed the term "Multiple-Valued Logic" meaning a logic with many logic-values. What are these logical values in the world of logic? Are they 'true', 'false', 'half-true', 'half-false' … etc.? Based on multiple-logical values, MVL views our world as a black and white but with different degrees. Post algebra picks up on this term and provides a mathematical tool to work with multiple-valued logic systems based on its axioms. The algebra does not have a natural concept, as the case in Boolean algebra or AOP, to derive its operators and theorems from. On the other hand, AOP is an algebra that has a solid natural concept from which AOP derived its operators and theorems. By the priority concept, AOP was able to discover many facts about MVL systems that Post algebra could not do. For example, AOP discovered

- that every z-radix system has a basic set of operators called prioritors whose number depends on system radix and is given by z!.
- that any MVL equation can have z! distinct forms.
- there exist z! representations for any MVL functions.
- there exist z! expansions for any MVL functions.
- the image-Scaling theorem to replace DeMorgan's Laws
- absorptions theorem-III to replace Kleene's laws

The priority concept is a universal and global concept that AOP did not found, designed or create but used as a natural resource to analyze, build and design digital systems. If God implemented this concept in all of his creations, then why not use it in man-made machines?

14 Conclusion and Expectations

AOP is characterized by its insights and the simplicity of its concepts, notations, and mathematical operations. It is a multi-valued multi-operational switching algebra and it is a generalization to the formal generalizations of binary and multi-valued switching algebras. We have shown that AOP in a z-radix system has z! binary operations called **prioritors**, has z! **conservative unary operators** and has $z^2(z-1)$ unary **orthogonal operators**. Further, we have shown: (1) the development of AOP from the priority concept and principle; (2) the TAS systems of AOP; (3) the intrinsic and extrinsic theorems AOP; (4) the **advanced theorems** of AOP: the image-scaling theorem, uniform degeneracy theorem, orthogonal theorem I, orthogonal theorem II, expansion theorem I, expansion theorem II; (5) (6) the proofs of the basic and advanced theorems of AOP; (4) the prioritors of the binary, ternary, and quaternary systems; (5) that Boolean and Post algebras are special cases of AOP.

Furthermore, we have shown: (1) how the uniform degeneracy theory of AOP extended the duality theory used by Boolean and Post algebras; (2) how a MVL equation can be degenerated into z! equations; (3) how the uniform degeneracy operation replaced the "dual" operation used by Boolean and Post algebras; (4) how the orthogonal theorems I & II of AOP extended the representations of MVL functions from two representations (sum-of-products and product-of-sums) to z! representations; (5) how the expansions theorems I&II of AOP extended the expansion of MVL functions from two expansions to z! expansions; (6) how the image-scaling theorem of AOP replaced DeMorgan's laws; (7) how the absorption theorem-III of AOP replaced Kleene's laws; (8) how Boolean, Post algebra and Kleenean algebras are special cases of AOP; (9) how AOP reduces MVL circuits complexity

Multiple-Operational Logic (MOL) is a new area that uses multiple-operators from unary and binary operators to design digital circuits. It is aimed at introducing, into logical systems, a variety of new operators that will make design more flexible than would be using just the MVL traditional operators. AOP is just a starting point in this field. AOP opens a new wide area for research. The large number of prioritors in various radii needs to be investigated more in terms of their use in digital circuits design**.** If researchers get interested in this field, then they can work toward the means that will develop the concepts of this field as they did for the field of MVL.

15 Tables

Table 6: Conservative Unary Operators

Binary System Conservative Unary Operators												
Unary		List By S-code				List By B's Code				Significant		Name, Function
No	α	α	α-	α*	α#	α	α-	α*	α#	MSD	LSD	
1	B1	2S01	2S01	2S10	2S01	B1	B1	B2	B1	0	1	NOT, 1-complement
2	B2	2S10	2S10	2S01	2S01	B2	B2	B1	B1	1	0	Identity, 0-Complement
AND=B1=Δ, OR=B2=∇, NOT=Δ=2S01, Δ=2S01, ∇=2S10												
Ternary System Conservative Unary Operators												
Unary		List By S-code				List By T's Code				Significant		Name, Function
No	α	α	α-	α*	α#	α	α-	α*	α#	MSD	LSD	
1	T1	3S012	3S012	3S210	3S012	T1	T1	T6	T1	0	2	MV-NOT, 2-complment
2	T2	3S021	3S102	3S120	3S201	T2	T3	T4	T5	0	1	Successor/up
3	T3	3S102	3S021	3S201	3S120	T3	T2	T5	T4	1	2	.
4	T4	3S120	3S120	3S021	3S201	T4	T4	T2	T5	1	0	0-complement
5	T5	3S201	3S201	3S102	3S120	T5	T5	T3	T4	2	1	1-complement
6	T6	3S210	3S210	3S012	3S012	T6	T6	T1	T1	2	0	Identity
MIN=T1=Δ, MAX=T6 =∇, MV-NOT=Δ=3S012, Δ=3S012, ∇=3S210												
Quaternary System Conservative Unary Operators												
Unary		List By S-code				List By Q's Code				Significant		Name, Function
No	α	α	α-	α*	α#	α	α-	α*	α#	MSD	LSD	
1	Q1	4S0123	4S0123	4S3210	4S0123	Q1	Q1	QO	Q1	0	3	MV-NOT, 3-Complement
2	Q2	4S0132	4S1023	4S2310	4S1032	Q2	Q7	QI	Q8	0	2	.
3	Q3	4S0213	4S0213	4S3120	4S0123	Q3	Q3	QM	Q1	0	3	.
4	Q4	4S0231	4S1203	4S1320	4S2301	Q4	Q9	QC	QH	0	1	.
5	Q5	4S0312	4S2013	4S2130	4S1032	Q5	QD	QG	Q8	0	2	.
6	Q6	4S0321	4S2103	4S1230	4S2301	Q6	QF	QA	QH	0	1	Successor/Up
7	Q7	4S1023	4S0132	4S3201	4S1032	Q7	Q2	QN	Q8	1	3	.
8	Q8	4S1032	4S1032	4S2301	4S0123	Q8	Q8	QH	Q1	1	2	.
9	Q9	4S1203	4S0231	4S3021	4S1032	Q9	Q4	QK	Q8	1	3	.
10	QA	4S1230	4S1230	4S0321	4S2301	QA	QA	Q6	QH	1	0	0-Complement
11	QB	4S1302	4S2031	4S2031	4S0123	QB	QE	QE	Q1	1	2	.
12	QC	4S1320	4S2130	4S0231	4S2301	QC	QG	Q4	QH	1	0	.
13	QD	4S2013	4S0312	4S3102	4S2301	QD	Q5	QL	QH	2	3	.
14	QE	4S2031	4S1302	4S1302	4S0123	QE	QB	QB	Q1	2	1	.
15	QF	4S2103	4S0321	4S3012	4S2301	QF	Q6	QJ	QH	2	3	Predecessor/Down
16	QG	4S2130	4S1320	4S0312	4S1032	QG	QC	Q5	Q8	2	0	.
17	QH	4S2301	4S2301	4S1032	4S0123	QH	QH	Q8	Q1	2	1	1-Complement
18	QI	4S2310	4S2310	4S0132	4S1032	QI	QI	Q2	Q8	2	0	.
19	QJ	4S3012	4S3012	4S2103	4S2301	QJ	QJ	QF	QH	3	2	2-Complement
20	QK	4S3021	4S3102	4S1203	4S1032	QK	QL	Q9	Q8	3	1	.
21	QL	4S3102	4S3021	4S2013	4S2301	QL	QK	QD	QH	3	2	.
22	QM	4S3120	4S3120	4S0213	4S0123	QM	QM	Q3	Q1	3	0	.
23	QN	4S3201	4S3201	4S1023	4S1032	QN	QN	Q7	Q8	3	1	.
24	QO	4S3210	4S3210	4S0123	4S0123	QO	QO	Q1	Q1	3	0	Identity
MIN=Q1=Δ, MAX=QO=∇, MV-NOT=Δ=4S0123, Δ=4S0123, ∇=4S3210												

Table 7: Orthogonal Operators in Binary, Ternary, and Quaternary Systems

Quaternary System								Ternary				Binary	
1	4Ω001	13	4Ω101	25	4Ω201	37	4Ω301	1	3Ω001	13	3Ω201	1	2Ω001
2	4Ω002	14	4Ω102	26	4Ω202	38	4Ω302	2	3Ω002	14	2Ω202	2	2Ω010
3	4Ω003	15	4Ω103	27	4Ω203	39	4Ω303	3	3Ω010	15	3Ω210	3	2Ω101
4	4Ω010	16	4Ω110	28	4Ω210	40	4Ω310	4	3Ω012	16	3Ω212	4	2Ω110
5	4Ω012	17	4Ω112	29	4Ω212	41	4Ω312	5	3Ω020	17	3Ω220		
6	4Ω013	18	4Ω113	30	4Ω213	42	4Ω313	6	3Ω021	18	3Ω221		
7	4Ω020	19	4Ω120	31	4Ω220	43	4Ω320	7	3Ω101				
8	4Ω021	20	4Ω121	32	4Ω221	44	4Ω321	8	3Ω102				
9	4Ω023	21	4Ω123	33	4Ω223	45	4Ω323	9	3Ω110				
10	4Ω030	22	4Ω130	34	4Ω230	46	4Ω330	10	3Ω112				
11	4Ω031	23	4Ω131	35	4Ω231	47	4Ω331	11	3Ω120				
12	4Ω032	24	4Ω132	36	4Ω232	48	4Ω332	12	3Ω121				

Table 8: Prioritors List In Binary, Ternary And Quaternary Digital Systems

Binary System Prioritors

Prioritor		List By The Priority-Assignment s-code				List By B's Code				Switches		STAS		Function Table
No	α	α	α-	α*	α#	α	α-	α*	α#	$\alpha^{\neg\wedge}$	$\alpha^{\neg\vee}$	α	α*	List By s-Code
1	B1	2S01	2S01	2S10	2S01	B1	B1	B2	B1	0	1	B1	B2	2S10:00 (AND)
2	B2	2S10	2S10	2S01	2S01	B2	B2	B1	B1	1	0	B2	B1	2S11:10 (OR)

AND=B1=Δ, OR=B2=∇, NOT=Δ=2S01, Δ=2S01, ∇=2S10

Ternary System Prioritors

Prioritor		List By The Priority-Assignment s-code				List By T's Code				Switches		STAS		Function Table
No	α	α	α-	α*	α#	α	α-	α*	α#	$\alpha^{\neg\wedge}$	$\alpha^{\neg\vee}$	α	α*	List By s-Code
1	T1	3S012	3S012	3S210	3S012	T1	T1	T6	T1	0	2	T1	T6	3S210:110:000 (MIN)
2	T2	3S021	3S102	3S120	3S201	T2	T3	T4	T5	0	1	T2	T4	3S220:210:000
3	T3	3S102	3S021	3S201	3S120	T3	T2	T5	T4	1	2	T3	T5	3S210:111:010
4	T4	3S120	3S120	3S021	3S201	T4	T4	T2	T5	1	0	T4	T2	3S212:111:210
5	T5	3S201	3S201	3S102	3S120	T5	T5	T3	T4	2	1	T5	T3	3S222:210:200
6	T6	3S210	3S210	3S012	3S012	T6	T6	T1	T1	2	0	T6	T1	3S222:211:210 (MAX)

MIN=T1=Δ, MAX=T6 =∇, MV-NOT=Δ=3S012, Δ=3S012, ∇=3S210

Quaternary System Prioritors

Prioritor		List By The Priority-Assignment s-code				List By Q's Code				Switches		STAS		Function Table
No	α	α	α-	α*	α#	α	α-	α*	α#	$\alpha^{\neg\wedge}$	$\alpha^{\neg\vee}$	α	α*	List By s-Code
1	Q1	4S0123	4S0123	4S3210	4S0123	Q1	Q1	QO	Q1	0	3	Q1	QO	4S3210:2210:1110:0000
2	Q2	4S0132	4S1023	4S2310	4S1032	Q2	Q7	QI	Q8	0	2	Q2	QI	4S3310:3210:1110:0000
3	Q3	4S0213	4S0213	4S3120	4S0123	Q3	Q3	QM	Q1	0	3	Q3	QM	4S3210:2220:1210:0000
4	Q4	4S0231	4S1203	4S1320	4S2301	Q4	Q9	QC	QH	0	1	Q4	QC	4S3230:2220:3210:0000
5	Q5	4S0312	4S2013	4S2130	4S1032	Q5	QD	QG	Q8	0	2	Q5	QG	4S3330:3210:3110:0000
6	Q6	4S0321	4S2103	4S1230	4S2301	Q6	QF	QA	QH	0	1	Q6	QA	4S3330:3220:3210:0000
7	Q7	4S1023	4S0132	4S3201	4S1032	Q7	Q2	QN	Q8	1	3	Q7	QN	4S3210:2210:1111:0010
8	Q8	4S1032	4S1032	4S2301	4S0123	Q8	Q8	QH	Q1	1	2	Q8	QH	4S3310:3210:1111:0010
9	Q9	4S1203	4S0231	4S3021	4S1032	Q9	Q4	QK	Q8	1	3	Q9	QK	4S3210:2212:1111:0210
10	QA	4S1230	4S1230	4S0321	4S2301	QA	QA	Q6	QH	1	0	QA	Q6	4S3213:2212:1111:3210
11	QB	4S1302	4S2031	4S2031	4S0123	QB	QE	QE	Q1	1	2	QB	QE	4S3313:3210:1111:3010
12	QC	4S1320	4S2130	4S0231	4S2301	QC	QG	Q4	QH	1	0	QC	Q4	4S3313:3212:1111:3210
13	QD	4S2013	4S0312	4S3102	4S2301	QD	Q5	QL	QH	2	3	QD	QL	4S3210:2222:1210:0200
14	QE	4S2031	4S1302	4S1302	4S0123	QE	QB	QB	Q1	2	1	QE	QB	4S3230:2222:3210:0200
15	QF	4S2103	4S0321	4S3012	4S2301	QF	Q6	QJ	QH	2	3	QF	QJ	4S3210:2222:1211:0210
16	QG	4S2130	4S1320	4S0312	4S1032	QG	QC	Q5	Q8	2	0	QG	Q5	4S3213:2222:1211:3210
17	QH	4S2301	4S2301	4S1032	4S0123	QH	QH	Q8	Q1	2	1	QH	Q8	4S3233:2222:3210:3200
18	QI	4S2310	4S2310	4S0132	4S1032	QI	QI	Q2	Q8	2	0	QI	Q2	4S3233:2222:3211:3210
19	QJ	4S3012	4S3012	4S2103	4S2301	QJ	QJ	QF	QH	3	2	QJ	QF	4S3333:3210:3110:3000
20	QK	4S3021	4S3102	4S1203	4S1032	QK	QL	Q9	Q8	3	1	QK	Q9	4S3333:3220:3210:3000
21	QL	4S3102	4S3021	4S2013	4S2301	QL	QK	QD	QH	3	2	QL	QD	4S3333:3210:3111:3010
22	QM	4S3120	4S3120	4S0213	4S0123	QM	QM	Q3	Q1	3	0	QM	Q3	4S3333:3212:3111:3210
23	QN	4S3201	4S3201	4S1023	4S1032	QN	QN	Q7	Q8	3	1	QN	Q7	4S3333:3222:3210:3200
24	QO	4S3210	4S3210	4S0123	4S0123	QO	QO	Q1	Q1	3	0	QO	Q1	4S3333:3222:3211:3210

MIN=Q1=Δ, MAX=QO=∇, MV-NOT=Δ=4S0123, Δ=4S0123, ∇=4S3210

Table 9: Number Of Prioritors In 2-31 Radices Systems

Radix	Prioritors Number	Radix	Prioritors Number
2	2	17	355,687,428,096,000
3	6	18	6,402,373,705,728,000
4	24	19	121,645,100,408,832,000
5	120	20	2,432,902,008,176,640,000
6	720	21	51,090,942,171,709,440,000
7	5,040	22	1,124,000,727,777,607,680,000
8	40,320	23	25,852,016,738,884,976,640,000
9	362,880	24	620,448,401,733,239,439,360,000
10	3,628,800	25	15,511,210,043,330,985,984,000,000
11	39,916,800	26	403,291,461,126,605,635,584,000,000
12	479,001,600	27	10,888,869,450,418,352,160,768,000,000
13	6,227,020,800	28	304,888,344,611,713,860,501,504,000,000
14	87,178,291,200	29	8,841,761,993,739,701,954,543,616,000,000
15	1,307,674,368,000	30	265,252,859,812,191,058,636,308,480,000,000
16	20,922,789,888,000	31	8,222,838,654,177,922,817,725,562,880,000,000

Table 10: Binary TAS systems

TAS	TASB1	TASB2
Base	2S01	2S10
ancestor	(B2,B1)	(B2,B2)
1	(B1,B2) B1	(B1,B1) B2
2	(B2,B1) B1	(B2,B2) B2

Table 11: Ternary TAS Systems

TAS	TAST1	TAST2	TAST3	TAST4	TAST5	TAST6
Base	3S012	3S021	3S102	3S120	3S201	3S210
ancestor	(T6,T1)	(T6,T2)	(T6,T3)	(T6,T4)	(T6,T5)	(T6,T6)
1	(T1,T6) T1	(T1,T5) T3	(T1,T4) T2	(T1,T3) T5	(T1,T2) T4	(T1,T1) T6
2	(T2,T4) T5	(T2,T3) T2	(T2,T6) T3	(T2,T5) T1	(T2,T1) T4	(T2,T2) T6
3	(T3,T5) T4	(T3,T6) T2	(T3,T2) T3	(T3,T1) T5	(T3,T4) T1	(T3,T3) T6
4	(T4,T2) T5	(T4,T1) T3	(T4,T5) T2	(T4,T6) T4	(T4,T3) T1	(T4,T4) T6
5	(T5,T3) T4	(T5,T4) T3	(T5,T1) T2	(T5,T2) T1	(T5,T6) T5	(T5,T5) T6
6	(T6,T1) T1	(T6,T2) T2	(T6,T3) T3	(T6,T4) T4	(T6,T5) T5	(T6,T6) T6

Table 12: Transferring The Function Table

Function Table				Term	Vectors Table			Representations	
#	u	v	f(u,v)	Type	Vector	Components		Terms Of	Terms Of
m	X_{m2}	X_{m1}	$F(X_m)$	✓×☑	X_m	X_{m2}	X_{m1}	Orthogonal-I	Orthogonal-II
1	0	0	0	✓	$X_1=00$	$X_{12}=0$	$X_{11}=0$	$(0\alpha u^{-\Delta 012}\, \alpha v^{-\Delta 012})\, \alpha^*$	$(u^{-\Delta 010}\, \alpha v^{-\Delta 010})\alpha^*$
2	0	1	1	×	$X_2=01$	$X_{22}=0$	$X_{21}=1$	*trivial*	*trivial*
3	0	2	2	☑	$X_3=02$	$X_{32}=0$	$X_{31}=2$	$(u^{-\Delta 012}\, \alpha v^{-\Delta 212})\alpha^*$	$(u^{-\Delta 012}\, \alpha v^{-\Delta 212})\alpha^*$
4	1	0	1	×	$X_4=10$	$X_{42}=1$	$X_{41}=0$	*trivial*	*trivial*
5	1	1	1	×	$X_5=11$	$X_{52}=1$	$X_{51}=1$	*trivial*	*trivial*
6	1	2	1	×	$X_6=12$	$X_{62}=1$	$X_{61}=2$	*trivial*	*trivial*
7	2	0	2	☑	$X_7=20$	$X_{72}=2$	$X_{71}=0$	$(u^{-\Delta 212}\, \alpha v^{-\Delta 012})\alpha^*$	$(u^{-\Delta 212}\, \alpha v^{-\Delta 012})\alpha^*$
8	2	1	1	×	$X_8=21$	$X_{82}=2$	$X_{81}=1$	*trivial*	*trivial*
9	2	2	2	☑	$X_9=22$	$X_{92}=2$	$X_{91}=2$	$(u^{-\Delta 212}\, \alpha v^{-\Delta 212})$	$(u^{-\Delta 212}\, \alpha v^{-\Delta 212})$
Orthogonal-I Rep: $f(u,v)= (0\alpha u^{-\Delta 012}\, \alpha v^{-\Delta 012})\alpha^*(u^{-\Delta 012}\, \alpha v^{-\Delta 212})\alpha^*(u^{-\Delta 212}\, \alpha v^{-\Delta 012})\alpha^*(u^{-\Delta 212}\, \alpha v^{-\Delta 212})$									
Orthogonal-II Rep: $f(u,v)= (u^{-\Delta 010}\, \alpha v^{-\Delta 010})\alpha^*(u^{-\Delta 012}\, \alpha v^{-\Delta 212})\alpha^*(u^{-\Delta 212}\, \alpha v^{-\Delta 012})\alpha^*(u^{-\Delta 212}\, \alpha v^{-\Delta 212})$									
×Trivial term, ✓ non-trivial term, ☑ NMRV-term, NMRV=2, MRV=1, $\alpha=T_3$, STAS=(T_3,T_5)									

Table 13: Uniform Image-Scaling Of α=Q1 Under f=4S3012

No	A	B	A^{-f}	B^{-f}	A α B	$A\alpha^{offf}B$	$(A\alpha\, B)^{-f}$	$A^{-f}\, \alpha^{offf}\, B^{-f}$
1	0	0	2	2	0	0	2	2
2	0	1	2	1	0	1	2	2
3	0	2	2	0	0	2	2	2
4	0	3	2	3	0	0	2	2
5	1	0	1	2	0	1	2	2
6	1	1	1	1	1	1	1	1
7	1	2	1	0	1	2	1	1
8	1	3	1	3	1	1	1	1
9	2	0	0	2	0	2	2	2
10	2	1	0	1	1	2	1	1
11	2	2	0	0	2	2	0	0
12	2	3	0	3	2	2	0	0
13	3	0	3	2	0	0	2	2
14	3	1	3	1	1	1	1	1
15	3	2	3	0	2	2	0	0
16	3	3	3	3	3	3	3	3

Table 14: The Uniform Degeneracy Of Prioritors

The Uniform Degeneracy Of Prioritors In Binary, Ternary, and Quaternary Systems												
Quaternary System					Ternary System				Binary System			
Q's	123456	789ABC	DEFGHI	JKLMNO								
	444444	444444	444444	444444	T's	12	34	56				
⇒	SSSSSS	SSSSSS	SSSSSS	SSSSSS	⇒	33	33	33	B's	1	2	
Select	000000	111111	222222	333333	Select	SS	SS	SS	⇒	2	2	
"f"	112233	002233	001133	001122	"f"	00	11	22	Select	S	S	
⇒	231312	230302	130301	120201	⇒	12	02	01	"f"	0	1	
	323121	323020	313010	212010		21	20	10	⇒	1	0	
α	The Uniform Degeneracy α^{offf}				α	α^{offf}			α	α^{offf}		
Q's	123456	789ABC	DEFGHI	JKLMNO	T's	12	34	56	B's	1	2	
Q1	OIMCGA	NHK6E4	LBJ582	F9D371	T1	64	52	31	B1	2	1	
Q2	NHLBF9	OIJ5D3	MCK671	GAE482	T2	53	61	42	B2	1	2	
Q3	MGOAIC	KEN4H6	J8L2B5	D7F193	T3	46	25	13				
Q4	LFN9HB	JDO3I5	K7M1C6	E8G2A4	T4	35	16	24				
Q5	KEJ8D7	MGL2F1	OAN493	ICH6B5	T5	21	43	65				
Q6	JDK7E8	LFM1G2	N9O3A4	HBI5C6	T6	12	34	56				
Q7	IOCMAG	HN6K4E	BL5J28	9F3D17								
Q8	HNBL9F	IO5J3D	CM6K17	AG4E28								
Q9	GMAOCI	EK4N6H	8J2L5B	7D1F39								
QA	FL9NBH	DJ3O5I	7K1M6C	8E2G4A								
QB	EK8J7D	GM2L1F	AO4N39	CI6H5B								
QC	DJ7K8E	FL1M2G	9N3O4A	BH5I6C								
QD	CAIGOM	64HENK	52B8LJ	3197FD								
QE	B9HFNL	53IDOJ	61C7MK	42A8GE								
QF	ACGIMO	46EHKN	258BJL	1379DF								
QG	9BFHLN	35DIJO	167CKM	248AEG								
QH	87EDKJ	21GFML	43A9ON	65CBIH								
QI	78DEJK	12FGLM	349ANO	56BCHI								
QJ	645231	CAB897	IGHEFD	OMNKLJ								
QK	536142	B9C7A8	HFIDGE	NLOJMK								
QL	462513	AC8B79	GIEHDF	MOKNJL								
QM	351624	9B7C8A	FHDIEG	LNJOKM								
QN	214365	87A9CB	EDGFIH	KJMLON								
QO	123456	789ABC	DEFGHI	JKLMNO								

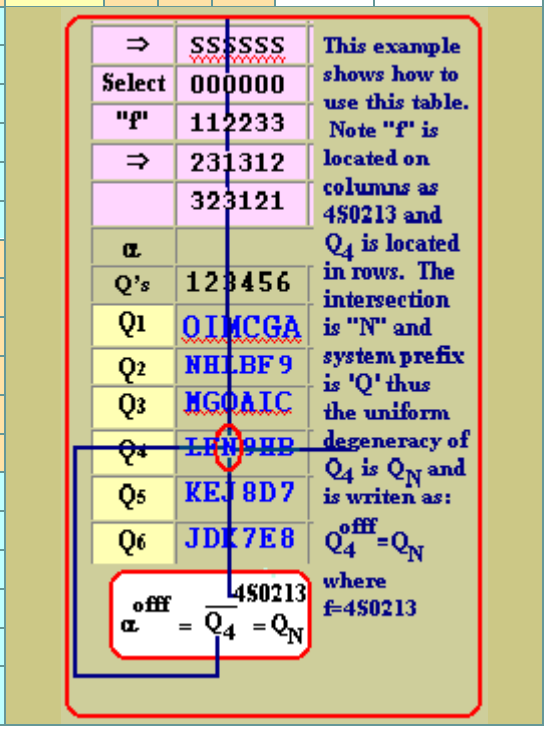


Table 15: Degenerate Equations Of Distribution Theorem

Degenerate Equations Of Example 1					
#	"offf"	Uniform Degeneracy Of $A\ \alpha_1\ (B\ \alpha_O C)=(A\ \alpha_1 B)\ \alpha_O(A\ \alpha_1 C)$ Under offf Operator	#	"offf"	Uniform Degeneracy Of $A\ \alpha_1\ (B\ \alpha_O C)=(A\ \alpha_1 B)\ \alpha_O(A\ \alpha_1 C)$ Under offf Operator
1	4S0123	$A\ \alpha_O\ (B\ \alpha_1 C)=(A\ \alpha_O\ B)\ \alpha_1(A\ \alpha_O C)$	13	4S2013	$A\ \alpha_L\ (B\ \alpha_D C)=(A\ \alpha_L B)\ \alpha_D(A\ \alpha_L C)$
2	4S0132	$A\ \alpha_I\ (B\ \alpha_2\ C)=(A\ \alpha_I\ B)\ \alpha_2\ (A\ \alpha_I C)$	14	4S2031	$A\ \alpha_B\ (B\ \alpha_E C)=(A\ \alpha_B B)\ \alpha_E(A\ \alpha_B\ C)$
3	4S0213	$A\ \alpha_M\ (B\ \alpha_3\ C)=(A\ \alpha_M B)\ \alpha_3(A\ \alpha_M C)$	15	4S2103	$A\ \alpha_J\ (B\ \alpha_F C)=(A\ \alpha_J B)\ \alpha_F(A\ \alpha_J\ C)$
4	4S0231	$A\ \alpha_C\ (B\ \alpha_4 C)=(A\ \alpha C_B)\ \alpha_4(A\ \alpha_C C)$	16	4S2130	$A\ \alpha_5\ (B\ \alpha_G C)=(A\ \alpha_5 B)\ \alpha_G(A\ \alpha_5\ C)$
5	4S0312	$A\ \alpha_G\ (B\ \alpha_5\ C)=(A\ \alpha_G\ B)\ \alpha_5(A\ \alpha_G C)$	17	4S2301	$A\ \alpha_8\ (B\ \alpha_H C)=(A\ \alpha_8 B)\ \alpha_H(A\ \alpha_8\ C)$
6	4S0321	$A\ \alpha_A\ (B\ \alpha_6\ C)=(A\ \alpha_A\ B)\ \alpha_6(A\ \alpha_A\ C)$	18	4S2310	$A\ \alpha_2\ (B\ \alpha_I C)=(A\ \alpha_2 B)\ \alpha_I(A\ \alpha_2\ C)$
7	4S1023	$A\ \alpha_N\ (B\ \alpha_7\ C)=(A\ \alpha_N\ B)\ \alpha_7(A\ \alpha_N\ C)$	19	4S3012	$A\ \alpha_F\ (B\ \alpha_J C)=(A\ \alpha_F B)\ \alpha_J(A\ \alpha_F\ C)$
8	4S1032	$A\ \alpha_H\ (B\ \alpha_8\ C)=(A\ \alpha_H\ B)\ \alpha_8(A\ \alpha_H\ C)$	20	4S3021	$A\ \alpha_9\ (B\ \alpha_K C)=(A\ \alpha_9 B)\ \alpha_K(A\ \alpha_9\ C)$
9	4S1203	$A\ \alpha_K\ (B\ \alpha_9\ C)=(A\ \alpha_K\ B)\ \alpha_9(A\ \alpha_K\ C)$	21	4S3102	$A\ \alpha_D\ (B\ \alpha_L C)=(A\ \alpha_D B)\ \alpha_L(A\ \alpha_D\ C)$
10	4S1230	$A\ \alpha_6\ (B\ \alpha_A\ C)=(A\ \alpha_6\ B)\ \alpha_A(A\ \alpha_6\ C)$	22	4S3120	$A\ \alpha_3\ (B\ \alpha_M C)=(A\ \alpha_3 B)\ \alpha_M(A\ \alpha_3\ C)$
11	4S1302	$A\ \alpha_E\ (B\ \alpha_B\ C)=(A\ \alpha_E\ B)\ \alpha_B(A\ \alpha_E\ C)$	23	4S3201	$A\ \alpha_7\ (B\ \alpha_N C)=(A\ \alpha_7 B)\ \alpha_N(A\ \alpha_7\ C)$
12	4S1320	$A\ \alpha_4\ (B\ \alpha_C\ C)=(A\ \alpha_4\ B)\ \alpha_C(A\ \alpha_4\ C)$	24	4S3210	$A\ \alpha_1\ (B\ \alpha_O C)=(A\ \alpha_1 B)\ \alpha_O(A\ \alpha_1\ C)$

16 Figures

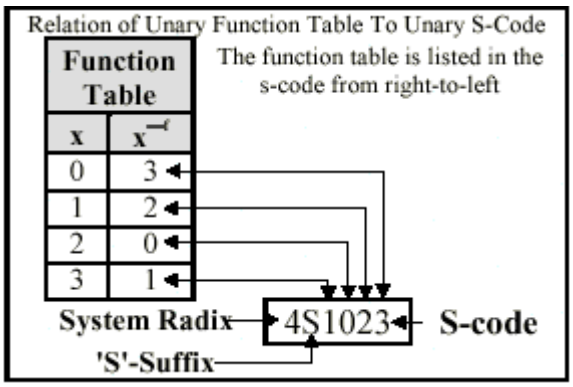


Figure 5: Unary S-Code Format

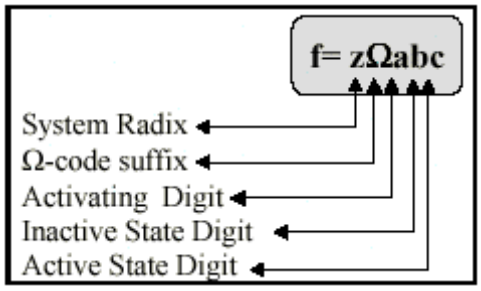


Figure 6: Orthogonal Code Format

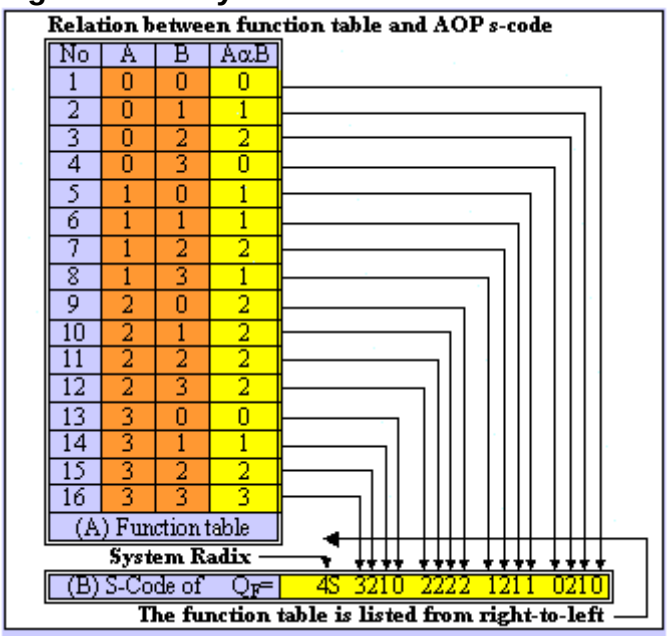


Figure 7: Prioritors S-Code Format

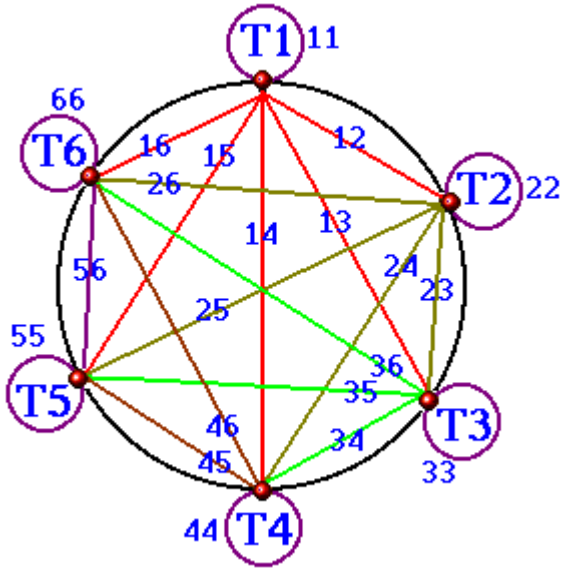


Figure 8: Map Of All Possible Pairs of Ternary Prioritors

17 BIBLIOGRAPHY

1. S.L. Hurst, "**Multiple-Valued Logic-Its Status and its Future**, " *IEEE trans. computers*, Vol. C-33, No 12, pp.1160-1179, DEC 1984.

2. J. T. Butler, "**Multiple-Valued Logic in VLSI Design,** *" IEEE Computer Society Press Technology Series*, 1991.

3. C.M. Allen, D.D. Givone "**The Allen-Givone Implementation Oriented Algebra,** " in *Computer Science and Multiple-Valued Logic: Theory and Applications*, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 268-288.

4. G. Epstein, "**The Lattice Theory Of Post algebras,** " in *Computer Science and Multiple-Valued Logic: Theory and Applications*, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 23-40.

5. I.G. Rosenberg, "**Completeness Properties Of Multiple-Valued Logic Algebras,** " in *Computer Science and Multiple-Valued Logic: Theory and Applications*, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, NY, 1984. pp. 150-174.

6. G. Abraham, "**Multiple-Valued Negative Resistance Integrated Circuits,** " in *Computer Science and Multiple-Valued Logic: Theory and Applications*, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 394-446.

7. G. Epstein, G. Frieder and D.C. Rine, **"The Development Of Multiple-Valued Logic As Related To Computer Science,** " in *Computer Science and Multiple-Valued Logic: Theory and Applications*, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 87-107.

8. J.C. Muzio, T.C. Wesselkamper, "**Multiple-Valued Switching Theory,**" Adam Hilger, Boston, Mass., 1986.

9. D.C. Rine, "**Computer Science And Multiple-Valued Logic: Theory And Applications,** " second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984.

10. K.C. Smith, "**The Prospects Of Multiple-Valued Logic: A Technology And Application View, "** *IEEE Transaction on Computers*, pp 619-632, DEC 1981.

11. G. Epstein, A. Horn, "**Chain Based Lattices,** " in Computer Science and Multiple-Valued Logic: Theory and Applications, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 58-76.

12. Y. Hata, K. Nakashima and K. Yamato, "**Some Fundamental Properties Of Multiple-Valued Kleenean Functions And Determination Of Their Logic Formulas**, " *IEEE Trans. Computers.;* Vol 42, No 8, pp 950-961, AUG 1993.

13. George Epstein, Alfred Horn, "**P-Algebras, An Abstraction From Post algebras, "** in Computer Science and Multiple-Valued Logic: Theory and Applications, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 108-120.

14. Grimaldi, Ralph, "**Discrete And Combinatorial Mathematics: An Applied Introduction, "** 2nd ed., Addison-Wesley Publishing Company, 1989.

15. K. Smith, "**Multiple-Valued Logic: Tutorial And Appreciation, "** *IEEE Transaction on Computers*, pp 17-27, April 1988.

16. Stephen Su, Peter T., "**Computer Simplification Of Multi-Valued Switching Functions,**" in Computer Science and Multiple-Valued Logic: Theory and Applications, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 195-226.

17. William R. Smith, "**Minimization Of Multi-Valued Functions,** " in Computer Science and Multiple-Valued Logic: Theory and Applications, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. .227-267.

18. Josep M., Ventura V., "**Abstract Characterization Of Four-Valued Logic**," IEEE Int. Symp. Multiple-Valued logic, 1988, p.389-396.

19. G. Epstein and R.R Loka, "**Almost Orthogonal Functions, "** IEEE Int. Symp. Multiple-Valued logic, 1988, p.405-411.

20. Iov G. Rosenberg, Dan A. Simovici, "**Algebraic Aspects Of Multiple-Valued Logic, "** IEEE Int. Symp. Multiple-Valued logic, 1988, p.266-275.

21. Yoshifumi Tsuchiya, "**Four-Valued Logic Using Two Lines And Its Applications To Model Logic,"** IEEE Int. Symp. Multiple-Valued logic, 1988, p.398-404.

22. P. Garcia, E. Esteva, "**Representation Theorem Of Ockham Algebras, "** IEEE Int. Symp. Multiple-Valued logic, 1989, p.14-19.

23. T. Traczyk, "**Post algebras Through P**0 **and P**1 **lattices,** " in Computer Science and Multiple-Valued Logic: Theory and Applications, D.C. Rine, second edition, D.C. Rine, ed., The Elsevier North-Holland, New York, N.Y., 1984. pp. 121-142.

24. T. Sasao, "**Multiple-Valued Decomposition of Generalized Boolean Functions and the Complexity of Programmable Logic Arrays,**" IEEE Trans. Computers, Sept. 1981, pp. 635-643.