K.A.U.: Sci., vol. 2, pp. 117-136 (1410 A.H. / 1990 A.D.)

CITALOG
Compact and Integrated Tasim Logic Closure

FEVZİ ÜNLÜ
Department of Computer Science,
King Abdulaziz University,
Jeddah, Kingdom of Saudi Arabia.

ABSTRACT. A two-valued logic software minimizing, Compact and Integrated TASIM Logic (CITALOG) closure has been developed for realizing logical operators and Boolean functions of any finite number of operands and any finite number of variables under the well-defined construction and operating rules of TASIM in order to find the self minimizing software designs of the hardware structures in the Logic Design.

Introduction

Regular expressions and their finite-state transducers are two important notions in the theory of computation[1]. The notion of classical logic is well known by the references of [1, 2, 5, 6] recorded in Ref.[2]. We know that over the last two or three decades, computer hardware has undergone dramatic cost reduction by the application of the classical logic to the fundamental parts of hardware design[3]. This has not been accompanied by corresponding reductions in the software cost of computing systems. Software developments still takes ~ 75% of the total computing system budget in our time, compared with 5% in the 1950's. Hence, we need a new technique for software minimization. For this reason, a Tidy Automatic Sequential Information-processing Mechanism, which is called TASIM, was developed[2,4]. It is a functional high-level formal language for creating and realizing TASIM definable regular expressions and their finite-state transducers as software in a TASIM closure tC. Some logic operators in the closure of unary and binary logic operators were realized by TASIM[2,4,5]. Some algebraic data structures were studied in terms of TASIM[6]. A notion of TASIM logic was developed and used for the realization of Boolean

Permanent address: Kazim Dirik Mahallesi, Mustafa Kemal Caddesi, 231 Sokak No. 2/6, Bornova, Izmir, Turkey.

Algebra. Syntax, semantics and pragmatics of a TASIM closure tC over an alphabet AA are formalized[7]. TASIM logic realizations of (a) a generalized TASIM multiplex, (b) compact and integrated TASIM flip-flop closures for software designs of hardware structures, and (c) TASIM storage closure were developed[8]. The notion of high level programming languages can be found in Ref.[9].

In this paper, the author is studying a two-valued logical software minimizing, Compact and Integrated TASIM Logic (CITALOG) closure %tC as a subset of a TASIM closure tC over an alphabet AA for realizing logical operators of any finite number of operands, and Boolean functions of any finite number of variables in order to find the software designs of hardware structures in the logic design. For this, all necessary rules of minimizations were introduced into TASIM for minimizing a TASIM program automatically in The %tC.

Four sections follow this section. In section two we introduce the notations used in the article and give their meanings. In section three we present the notion of CITALOG closure. Section four is on applications. Section five is on the conclusion. An appendix is added to this paper. One can find a table for all logic operators in the unary operator closure, and also a table for all logic operators in the binary logic operator closure of the Logic Design in this appendix.

Notations Used and Their Meanings

$\subset$	Proper subset symbol. If $A \subset B$, then A is a proper subset of B.
$\subseteq$	Subset symbol. If $A \subseteq B$, then A is a subset of B.
$<$	Symbol of "less than" operator.
$\leq$	Symbol of "equal or less than" operator.
$>$	Symbol of "greater than" operator.
$\geq$	Symbol of "greater or equal" operator.
$<\ldots>$	String terminators in *BNF* grammar.
P	The set of positive numbers, $P = \{1, 2, \ldots\}$.
$P[n]$	The set of positive numbers up to n, $P[n] = \{1, 2, \ldots, n\}$.
N	The set of natural numbers, $N = \{0, 1, 2, \ldots\}$.
$N[n]$	The set of natural numbers up to n, $N[n] = \{0, 1, \ldots, n\}$.
AA	*A* none empty and finite alphabet of TASIM closure, where $AA := < aS, aL, aLC[n], aT > = <aS> \mid <aL> \mid <aLC[n]> \mid <aT>$.
aS, $\%aS$	None empty and finite alphabets of special symbols such that $\%aS \subseteq aS$.
aL, $\%aL$	None empty and finite alphabets of letters such that $\%aL \subseteq aL$.
$aLC[k]$, $\%aLC[j]$	Positive closures of aL and $\%aL$ up to length k and j.
aT, $\%aT$	Finite alphabets of special reserved TASIM-names such that $\%aT \subseteq aT$.
tC, $\%tC$	TASIM closure and CITALOG closure on an alphabet AA such that $\%tC \subseteq tC$.

V	A special symbol in aT for representing the notion of a 2-valued ivasmode (initial and final value space mode).
(	Open parenthesis used as a special symbol in aS.
)	Close parenthesis used as a special symbol in aS.
)	A pair of parentheses used as a binary symbol for grouping the construction levels of a TASIM. It is also used as a generalized instruction-operator in the TASIM closure tC.
	A special symbol in aS for the complete substitution operator.
	A special symbol in aS for the reduction and expansion operator in a TASIM closure tC, in the sense of equivalence.
u	*A* special symbol in aS for supervising a variable in a functional TASIM in tC.
ux	A special string called as supervisor in a functional TASIM $ux.G$, where u is a supervisor and supervises x by making it a dependent variable in a TASIM functional body G.
G	A pseudo symbol for representing the body of a functional TASIM.
S	A special symbol in aS for representing the generalized substitution operator in a TASIM closure tC.
	The generalized substitution operator for substituting each independent x by y in a TASIM appearing after in the same level with $xSy@$, where '@' is a special symbol in aS for representing empty word separator.
$0:F$	A special symbol in aT for representing the notion of the falsity in the human thought and it has a functional TASIM representation $ux[1]ux[2].x[2]$ in this paper.
	A special symbol in aT for representing the notion of the truth in the human thought and it has a functional TASIM representation $ux[1]ux[2].x[1]$ in this paper.
$sG[i], G[i]$	Two instructional TASIMs were used for representing the bodies of some functional TASIMs in the definition of a CITALOG closure. They are logical variables taken from $aLC[n]$ and assume functional TASIMs as values in a TASIM closure tC.
$U[n]$	An n-ary and two-valued CITALOG operator in $\%tC$ for n is in P.
$UC[n]$	CITALOG operator closure in $\%tC$ for n is in P.
$I[n]$	An n-ary instruction in the two-valued CITALOG closure $\%tC$ for n is in P.
	Two-valued CITALOG instruction closure in $\%tC$ for n is in P.
CITALOG	Compact and Integrated TAsim LOGic.
$x[n]$	The nth CITALOG variable for n is in P.
$x[n+1]$	The $[n+1]$th CITALOG variable for n is in P.

Two-Valued CITALOG Closure

In this section we define a two-valued self minimizing 'Compact and Integrated TASIM Logic (CITALOG) closure' and study its properties.

Definition 1

Let *tC* be a TASIM closure on an alphabet *AA* = <*aS*, *aL*, *aLC*[*k*], *aT*> and %*tC* be a TASIM closure on an alphabet %*AA* = <%*aS*, %aL, %*aLC*[*j*], %*aT*> such that $j \leq k$, %*aS* ⊆ *aS*, %*aL* ⊆ *aL*, %*aLC*[*j*] ⊆ *aLC*[*k*] and %*aT* ⊆ *aT*. If %*tC* satisfies the following laws then it is called a two-valued CITALOG closure:

*L*1: A two-valued CITALOG closure has two distinct elements *F* and *T* in %*aT* implies that *F* and *T* are in *aT*.

*L*2: A two-valued CITALOG closure has realizations of the two distinct elements *F* and *T* in an ivasmode *V* as *V* = { *F*:= *ux*[1]*ux*[2].*x*[2], *T*:= *ux*[1]*ux*[2].*x*[1] } such that *V* ⊆ %*tC* implies that *V* ⊆ *tC* for *x*[1] and *x*[2] are two TASIM variables in %*aLC*[*j*] and *u* is a supervisor in %*aS*.

*L*3: Let *x*[0], *x*[1], ..., *x*[*n*] be *n* + 1 TASIM variables taken from %*aLC*[*j*] and *c*[0], *c*[1], ..., *c*[*m*], $m = 2^n - 1$ be names for constant data objects also taken from %*aLC*[*j*] assuming values on *V*.

(a) (*B*1) If *G*[1] = ((*x*[1]@*c*[1])*c*[0]) then *U*[1] := *ux*[1].*G*[1] is called a unary CITALOG operator in %*tC*.

(*B*2) If *sG*[*n*–1] = *c*[*i*]*Sc*[*ii*]@*G*[*n*–1] for $n \geq 2$ and all *c*[*i*] in *G*[*n*–1], then *U*[*n*] = *ux*[*n*] ... *ux*[2]*ux*[1].*G*[*n*] is called an *n*-ary and two-valued CITALOG operator in %*tC*. Where $ii = i + 2^{n-1}$ and *G*[*n*] = ((*x*[*n*]@*sG*[*n*–1]) *G*[*n*–1]).

(b) *UC*[*n*] = { *U*[*n*] : *n* is in *P* } is called a two-valued CITALOG operator closure in %*tC*.

(c) *UC*[*n*] in %*tC* implies that *UC*[*n*] is in *tC*.

*L*4: Let *y*[1], *y*[2], ..., *y*[*n*] be *n* variable taken from %*aLC*[*j*] and used for naming software data objects that contain CITALOG creating or representing propositional TASIMs on the ivasmode *V* ⊂ %*tC*, for *n* is in *P*.

(a) (i) *I*[1]) = (*U*[1]@*y*[1]) = (*ux*[1]·*G*[1]@*y*[1]) = *x*[1]*Sy*[1]@*G*[1] is called a unary instruction in the two-valued CITALOG closure %*tC* for *n* = 1.

(ii) *I*[*n*] = ((...(*U*[*n*]@*y*[1])*y*[2] ... *y*[n–1])*y*[*n*]) = ((...(*ux*[*n*]*ux*[*n*–1]... *ux*[2]*ux*[1].*G*[*n*].@*y*[1])*y*[2]...*y*[*n*–1])*y*[*n*]) = *x*[*n*]*Sy*[*n*]@*x*[*n*–1]*Sy*[*n*–1] @...*x*[1] *Sy*[1]@*G*[*n*] is called an *n*-ary instruction in the two-valued CITALOG closure %*tC* for $n \geq 2$.

(b) *IC*[*n*] = { *I*[*n*] : *n* is in *P* } is called a two-valued CITALOG instruction closure in %*tC*.

(c) *IC*[*n*] is in %*tC* implies that *IC*[*n*] is in *tC*.

Theorem 1 (Fundamental Theorem 1)

Let $\%tC$ be a CITALOG closure. If Q and R any two arbitrary TASIMs in tC which do not contain the bodies of T and F in V as independent variables, then :

(a) $((T@R)Q) = R$,
(b) $((F@R)Q) = Q$.

Proof

Let $\%tC$ be a CITALOG closure. Let Q and R be two arbitrary TASIMs in tC satisfying conditions in the hypothesis of the theorem, then :

$$\begin{aligned}
\text{(a)}\ ((T@R)Q) &= ((ux[1]ux[2].x[1]@R)Q)\\
&= x[1]SR@x[2]SQ@x[1]\\
&= R,\\
\text{(b)}\ ((F@R)Q) &= ((ux[1]ux[2]\cdot x[2]@R)Q)\\
&= x[1]SR@x[2]SQ@x[2]\\
&= Q.
\end{aligned}$$

Theorem 2 (Fundamental Theorem 2)

Let $\%tC$ be a two-valued CITALOG closure. If x and y are two arbitrary CITALOG variables on the two-valued ivasmode $V = \{F, T\}$ and x' and y' are their complements on V, then the following self minimizings exist in the two-valued CITALOG closure $\%tC$.

(1) $((x@F)F) = F$,
(2) $((x@F)T) = x'$,
(3) $((x@F)x) = F$,
(4) $((x@F)x') = x'$,
(5) $((x@T)F) = x$,
(6) $((x@T)T) = T$,
(7) $((x@T)x) = x$,
(8) $((x@T)x') = x'$,
(9) $((x@x)F) = x$,
(10) $((x@x)T) = T$,
(11) $((x@x)x) = x$,
(12) $((x@x)x') = T$,
(13) $((x@x')F) = F$,
(14) $((x@x')T) = x'$,
(15) $((x@x')x) = F$,
(16) $((x@x')x') = x'$,
(17) $((x'@F)F) = F$,
(18) $((x'@F)T) = x$,
(19) $((x'@F)x) = x$,
(20) $((x'@F)x') = F$,
(21) $((x'@T)F) = x'$,

(22) $((x'@T)T) = T,$
(23) $((x'@T)x) = T,$
(24) $((x'@T)x') = x',$
(25) $((x'@x)F) = F,$
(26) $((x'@x)T) = x,$
(27) $((x'@x)x) = x,$
(28) $((x'@x)x') = F,$
(29) $((x'@x')F) = x',$
(30) $((x'@x')T) = T,$
(31) $((x'@x')x) = T,$
(32) $((x'@x')x') = x',$
(33) $((x@y)y) = y',$
(34) $((x@y')y') = y',$
(35) $((x'@y)y) = y,$
(36) $((x'@y')y') = y'.$

Proof

Let %*tC* be a CITALOG closure. Let x and y be two CITALOG variables on the ivasmode $V = \{T, F\}$ and x' and y' be their complements on V. Using the substitution rules of TASIM and the first Fundamental Theorem 1, we obtain :

(1) (i) For $x = F$, $((x@F)F) = ((F@F)F) = F.$
(ii) For $x = T$, $((x@F)F) = ((T@F)F) = F.$

Hence, $((x@F)F) = F$ is true for all possible values of x on V.

(2) (i) For $x = F$, $((x@F)T) = ((F@F)T) = T = x'$
(ii) For $x = T$, $((x@F)T) = ((T@F)T) = F = x'$

Hence, $((x@F)T) = x'$ is true for all possible values of x on V.

(3) (i) For $x = F$, $((x@F)x) = ((F@F)F) = F.$
(ii) For $x = T$, $((x@F)x) = ((T@F)F) = F.$

Hence, $((x@F)T) = F$ is true for all possible values of x on V.

(32) (i) For $x = F$, $((x'@x')x') = ((T@T)T) = T = x'.$
(ii) For $x = T$, $((x'@x')x') = ((F@F)F) = F = x'.$

Hence, $((x'@x')x') = x'$ is true for all possible values of x on V.

(33) (i) For $x = F, y = F$, $((x@y)y) = ((F@F)F) = F = y.$
(ii) For $x = F, y = T$, $((x@y)y) = ((F@T)T) = T = y.$
(iii) For $x = T, y = F$, $((x@y)y) = ((T@F)F) = F = y.$
(iv) For $x = T, y = T$, $((x@y)y) = ((T@T)T) = T = y.$

Hence, $((x@y)y) = y$ is true for all possible combinational values of x and y on V.

(36) (i) For $x = F$, $y = F$, $((x'@y')y') = ((T@T)T) = T = y'$.
(ii) For $x = F$, $y = T$, $((x'@y')y') = ((T@F)F) = T = y'$.
(iii) For $x = T$, $y = F$, $((x'@y')y') = ((F@T)T) = T = y'$.
(iv) For $x = T$, $y = T$, $((x'@y')y') = ((F@F)F) = F = y'$.

Hence, $((x'@y')y') = y'$ is true for all possible combinational values of x and y on V.

Theorem 3

Let $\%tC$ be a two-valued CITALOG closure with a two-valued CITALOG operator closure $UC[n]$ and a two-valued CITALOG instruction closure $IC[n]$ defined in terms of n variables $x[1], x[2], \ldots, x[n]$ on the ivasmode V and two constants values F and T of V. A two-valued n-ary CITALOG operator $U[n]$ in $UC[n]$ realizes all two-valued n-ary logic operators in the Logic Design on the ivasmode V and the Fundamental Theorems of two-valued CITALOG closure automatically minimizes them on V.

Proof

($B1$) For $n = 1$, $1U = ux[1].G[1] = ux[1].((x[1]@c[1])c[0])$. There are 4 possible two-valued unary operators in the Logic Design. One can observe them like in Table 1 of the appendix. $U[1]$ realizes and the Fundamental Theorems of two-valued CITALOG automatically minimizes :

(1) $1U[1]$ for $c[0] = F$ and $c[1] = F$,
(2) $1U[2]$ for $c[0] = T$ and $c[1] = F$,
(3) $1U[3]$ for $c[0] = F$ and $c[1] = T$,
(4) $1U[4]$ for $c[0] = T$ and $c[1] = T$,

on the ivasmode V.

Because, by substitution and using the Fundamental Theorems of two-valued CITALOG closure :

(1) $1U[1] = ux[1].((x[1]@F)F) = ux[1].F$ is obtained for $c[0] = F$ and $c[1] = F$, and
(i) $(1U[1]@F) = F$,
(ii) $(1U[1]@T) = F$.

(2) $1U[2] = ux[1].((x[1]@F)T) = ux[1].x[1]'$ is obtained for $c[0] = T$ and $c[1] = F$, and
(i) $(1U[2]@F) = T$,
(ii) $(1U[2]@T) = F$.

(3) $1U[3] = ux[1].((x[1]@T)F) = ux[1].x[1]$ is obtained for $c[0] = F$ and $c[1] = T$. ınd
(i) $(1U[3]@F) = F$,
(ii) $(1U[3]@T) = T$.

(4) $1U[4] = ux[1].((x[1]@T)T) = ux[1].T$ is obtained for $c[0] = c[1] = T$, and
(i) $(1U[4]@F) = T$,
(ii) $(1U[4]@T) = T$.

Hence, the $1U$ realizes all two-valued unary operators of the Logic Design on the ivasmode V as one can observe their definition like in Table 1 of the appendix and the Fundamental Theorems of CITALOG closure automatically minimizes them.

($B2$) For $n = 2$,

$$\begin{aligned} 2U &= ux[2]ux[1].G[2] \\ &= ux[2]ux[1].((x[2]@sG[1])G[1]) \\ &= ux[2]ux[1].((x[2]@((x[1]@c[3])c[2]))((x[1]@c[1])c[0])), \end{aligned}$$

and there are 16 possible two-valued binary operators in the Logic Design. One can observe them like in Table 2 of the appendix. $2U$ realizes and the Fundamental Theorems of two-valued CITALOG closure automatically minimizes :

(1) $2U[1]$ for $c[0] = F$, $c[1] = F$, $c[2] = F$, and $c[3] = F$,
(2) $2U[2]$ for $c[0] = T$, $c[2] = F$, $c[3] = F$, and $c[4] = F$,
(3) $2U[3]$ for $c[0] = F$, $c[1] = T$, $c[3] = F$, and $c[4] = F$,
(4) $2U[4]$ for $c[0] = T$, $c[1] = T$, $c[2] = F$, and $c[3] = F$,
(5) $2U[5]$ for $c[0] = F$, $c[1] = F$, $c[2] = T$, and $c[3] = F$,
(6) $2U[6]$ for $c[0] = T$, $c[1] = F$, $c[2] = T$, and $c[3] = F$,
(7) $2U[7]$ for $c[0] = F$, $c[1] = T$, $c[2] = T$, and $c[3] = F$,
(8) $2U[8]$ for $c[0] = T$, $c[1] = T$, $c[2] = T$, and $c[3] = F$,
(9) $2U[9]$ for $c[0] = F$, $c[1] = F$, $c[2] = F$, and $c[3] = T$,
(10) $2U[10]$ for $c[0] = T$, $c[1] = F$, $c[2] = F$, and $c[3] = T$,
(11) $2U[11]$ for $c[0] = F$, $c[1] = T$, $c[2] = F$, and $c[3] = T$,
(12) $2U[12]$ for $c[0] = T$, $c[1] = T$, $c[2] = F$, and $c[3] = T$,
(13) $2U[13]$ for $c[0] = F$, $c[1] = F$, $c[2] = T$, and $c[3] = T$,
(14) $2U[14]$ for $c[0] = T$, $c[1] = F$, $c[2] = T$, and $c[3] = T$,
(15) $2U[15]$ for $c[0] = F$, $c[1] = T$, $c[2] = T$, and $c[3] = T$,
(16) $2U[16]$ for $c[0] = T$, $c[1] = T$, $c[2] = T$, and $c[3] = T$,
on the ivasmode V.

Because, by substitution and using the Fundamental Theorems of two-valued CITALOG closure :

$$\begin{aligned} (1)\ 2U[1] &= ux[2]ux[1].((x[2]@((x[1]@F)F))((x[1]@F)F)) \\ &= ux[2]ux[1].((x[2]@F)F) = ux[2]ux[1].F \end{aligned}$$

is obtained for $c[0] = c[1] = c[2] = c[3] = F$, and

(i) for $x[2] = F$ and $x[1] = F$,
$((2U[1]@F)F) = ((F@F)F) = F$,
(ii) for $x[2] = F$ and $x[1] = T$,
$((2U[1]@F)T) = ((T@F)F) = F$,
(iii) for $x[2] = T$ and $x[1] = F$,
$((2U[1]@T)F) = ((F@F)F) = F$,

(iv) for $x[2] = T$ and $x[1] = T$,
$((2U[1]@T)T) = ((T@F)F) = F$.

(2) $2U[2] = ux[2]ux[1].((x[2]@((x[1]@F)F))((x[1]@F)T))$
$= ux[2]ux[1].((x[2]@F)x[1]')$

is obtained for $c[0] = T$, $c[1] = c[2] = c[3] = F$, and

(i) $((2U[2]@F)F) = ((F@F)T) = T$,
(ii) $((2U[2]@F)T) = ((T@F)T) = F$,
(iii) $((2U[2]@T)F) = ((F@F)F) = F$,
(iv) $((2U[2]@T)T) = ((T@F)F) = F$.

(3) $2U[3] = ux[2]ux[1].((x[2]@((x[1]@F)F))((x[1]@T)F))$
$= ux[2]ux[1].((x[2]@F)x[1])$

is obtained for $c[0] = F$, $c[1] = T$, $c[2] = c[3] = F$, and

(i) $((2U[3]@F)F) = ((F@F)F) = F$,
(ii) $((2U[3]@F)T) = ((F@F)T) = T$,
(iii) $((2U[3]@T)F) = ((T@F)F) = F$,
(iv) $((2U[3]@T)T) = ((T@F)T) = F$.

(4) $2U[4] = ux[2]ux[1].((x[2]@((x[1]@F)F))((x[1]@T)T))$
$= ux[2]ux[1].((x[2]@F)T)$

is obtained for $c[0] = c[1] = T$, $c[2] = c[3] = F$, and

(i) $((2U[4]@F)F) = ((F@F)T) = T$,
(ii) $((2U[4]@F)T) = ((F@F)T) = T$,
(iii) $((2U[4]@T)F) = ((T@F)T) = F$,
(iv) $((2U[4]@T)T) = ((T@F)T) = F$.

(5) $2U[5] = ux[2]ux[1].((x[2]@((x[1]@F)T))((x[1]@F)F))$
$= ux[2]ux[1].((x[2]@x[1]')F)$

is obtained for $c[0] = c[1] = F$, $c[2] = T$, $c[3] = F$, and

(i) $((2U[5]@F)F) = ((F@T)F) = F$,
(ii) $((2U[5]@F)T) = ((F@F)F) = F$,
(iii) $((2U[5]@T)F) = ((T@T)F) = T$,
(iv) $((2U[5]@T)T) = ((T@F)F) = F$.

(6) $2U[6] = ux[2]ux[1].((x[2]@((x[1]@F))((x[1]@F)T))$
$= ux[2]ux[1].((x[2]@x[1]')x[1]') = ux[2]ux[1].x[1]$

is obtained for $c[0] = T$, $c[1] = F$, $c[2] = T$, $c[3] = F$, and

(i) $((2U[6]@F)F) = ((F@T)T) = T$,
(ii) $((2U[6]@F)T) = ((F@F)F) = F$,
(iii) $((2U[6]@T)F) = ((T@T)T) = T$,
(iv) $((2U[4]@T)T) = ((T@F)F) = F$.

(7) $2U[7] = ux[2]ux[1].((x[2]@((x[1]@F)T))((x[1]@T)F))$
$= ux[2]ux[1].((x[2]@x[1]')x[1])$

is obtained for $c[0] = F$, $c[1] = T$, $c[2] = T$, $c[3] = F$, and

$$\begin{aligned}
&\text{(i)} && ((2U[7]@F)F) = ((F@T)F) = F,\\
&\text{(ii)} && ((2U[7]@F)T) = ((F@F)T) = F,\\
&\text{(iii)} && ((2U[7]@T)F) = ((T@T)F) = F,\\
&\text{(iv)} && ((2U[7]@T)T) = ((T@F)T) = F.
\end{aligned}$$

$$\begin{aligned}
(8)\ 2U[8] &= ux[2]ux[1].((x[2]@((x[1]@F)T))((x[1]@T)T))\\
&= ux[2]ux[1].((x[2]@x[1]')T)
\end{aligned}$$

is obtained for $c[0] = c[1] = c[2] = T$, $c[3] = F$, and

$$\begin{aligned}
&\text{(i)} && ((2U[8]@F)F) = ((F@T)T) = T,\\
&\text{(ii)} && ((2U[8]@F)T) = ((F@F)T) = T,\\
&\text{(iii)} && ((2U[8]@T)F) = ((T@T)F) = T,\\
&\text{(iv)} && ((2U[8]@T)T) = ((T@F)T) = F.
\end{aligned}$$

$$\begin{aligned}
(9)\ 2U[9] &= ux[2]ux[1].((x[2]@((x[1]@T)F))((x[1]@F)F))\\
&= ux[2]ux[1].((x[2]@x[1])F)
\end{aligned}$$

is obtained for $c[0] = c[1] = c[2] = F$, $c[3] = T$, and

$$\begin{aligned}
&\text{(i)} && ((2U[9]@F)F) = ((F@F)F) = F,\\
&\text{(ii)} && ((2U[9]@F)T) = ((F@T)F) = F,\\
&\text{(iii)} && ((2U[9]@T)F) = ((T@F)F) = F,\\
&\text{(iv)} && ((2U[9]@T)T) = ((T@T)F) = T.
\end{aligned}$$

$$\begin{aligned}
(10)\ 2U[10] &= ux[2]ux[1].((x[2]@((x[1]@T)F))((x[1]@F)T))\\
&= ux[2]ux[1].((x[2]@x[1])x[1]')
\end{aligned}$$

is obtained for $c[0] = T$, $c[1] = c[2] = F$, $c[3] = T$, and

$$\begin{aligned}
&\text{(i)} && ((2U[10]@F)F) = ((F@F)T) = T,\\
&\text{(ii)} && ((2U[10]@F)T) = ((F@T)F) = F,\\
&\text{(iii)} && ((2U[10]@T)F) = ((T@F)T) = F,\\
&\text{(iv)} && ((2U[10]@T)T) = ((T@T)F) = T.
\end{aligned}$$

$$\begin{aligned}
(11)\ 2U[11] &= ux[2]ux[1].((x[2]@((x[1]@T)F))((x[1]@T)F))\\
&= ux[2]ux[1].((x[2]@x[1])x[1]) = ux[1]ux[1].x[1]
\end{aligned}$$

is obtained for $c[0] = F$, $c[1] = T$, $c[2] = F$, $c[3] = T$, and

$$\begin{aligned}
&\text{(i)} && ((2U[11]@F)F) = ((F@F)F) = F,\\
&\text{(ii)} && ((2U[11]@F)T) = ((F@T)T) = T,\\
&\text{(iii)} && ((2U[11]@T)F) = ((T@F)F) = F,\\
&\text{(iv)} && ((2U[11]@T)T) = ((T@T)T) = T.
\end{aligned}$$

$$\begin{aligned}
(12)\ 2U[12] &= ux[2]ux[1].((x[2]@((x[1]@T)F))((x[1]@T)T))\\
&= ux[2]ux[1].((x[2]@x[1])T)
\end{aligned}$$

is obtained for $c[0] = c[1] = T$, $c[2] = F$, $c[3] = T$, and

$$\begin{aligned}
&\text{(i)} && ((2U[12]@F)F) = ((F@F)T) = T,\\
&\text{(ii)} && ((2U[12]@F)T) = ((F@T)T) = T,\\
&\text{(iii)} && ((2U[12]@T)F) = ((T@F)T) = F,\\
&\text{(iv)} && ((2U[12]@T)T) = ((T@T)T) = T.
\end{aligned}$$

$$\begin{aligned}(13)\ 2U[13] &= ux[2]ux[1].((x[2]@((x[1]@T)T))((x[1]@F)F))\\ &= ux[2]ux[1].x[2]\end{aligned}$$

is obtained for $c[0] = c[1] = F$, $c[2] = c[3] = T$, and

(i) $((2U[13]@F)T) = F$,
(ii) $((2U[13]@F)T) = F$,
(iii) $((2U[13]@T)F) = T$,
(iv) $((2U[13]@T)T) = T$.

$$\begin{aligned}(14)\ 2U[14] &= ux[2]ux[1].((x[2]@((x[1]@T)T))((x[1]@F)T))\\ &= ux[2]ux[1].((x[2]@T)x[1]')\end{aligned}$$

is obtained for $c[0] = T$, $c[1] = F$, $c[2] = c[3] = T$, and

(i) $((2U[14]@F)F) = ((F@T)T) = T$,
(ii) $((2U[14]@F)T) = ((F@T)F) = F$,
(iii) $((2U[14]@T)F) = ((T@T)T) = T$,
(iv) $((2U[14]@T)T) = ((T@T)F) = T$.

$$\begin{aligned}(15)\ 2U[15] &= ux[2]ux[1].((x[2]@((x[1]@T)T))((x[1]@T)F))\\ &= ux[2]ux[1].((x[2]@T)x[1])\end{aligned}$$

is obtained for $c[0] = F$, $c[1] = c[2] = c[3] = T$, and

(i) $((2U[15]@F)F) = ((F@T)F) = F$,
(ii) $((2U[15]@F)T) = ((F@T)T) = T$,
(iii) $((2U[15]@T)F) = ((T@T)F) = T$,
(iv) $((2U[15]@T)T) = ((T@T)T) = T$.

$$\begin{aligned}(16)\ 2U[16] &= ux[2]ux[1].((x[2]@((x[1]@T)T))((x[1]@T)T))\\ &= ux[2]ux[1].T\end{aligned}$$

is obtained for $c[0] = c[1] = c[2] = c[3] = T$, and

(i) $((2U[16]@F)F) = T$,
(ii) $((2U[16]@F)T) = T$,
(iii) $((2U[16]@T)F) = T$,
(iv) $((2U[16]@T)T) = T$.

Hence $U[2]$ realizes all possible two-valued binary operators of the Logic Design on the ivasmode V and the Fundamental Theorems of two-valued CITALOG closure automatically self minimizes them on the V.

Now one can complete the proof of the Theorem 3 as follow for any $U[n]$ by induction :

(I) Assume, for all $c[i]$ in $G[n-1]$, we define $sG[n-1] = c[i]Sc[ii]@G[n-1]$, $ii = i + 2^{n-1}$, and $G[n] = ((x[n]@sG[n-]) G[n-1])$ that $U[n-1] = ux[n-1] \ldots ux[2]ux[1].G[n-1]$ in $UC[n]$ realizes all two-valued $(n-1)$-ary operators and the Fundamental Théorems of two-valued CITALOG closure automatically self minimizes them is true for $n \geq 2$. Then, by induction on n in P, $ux[n] \ldots ux[2]ux[1].G[n]$ in $UC[n]$ realizes all two-valued n-ary operators for $G = ((x[n]@sG[n-1])G$

$[n-1]$) and the Fundamental Theorems of two-valued CITALOG closure automatically minimizes them is also true. Because the rth two-valued n-ary operators in the Logic Design has a truth table like Table 1, where each $c[i]$ in the column $nU[r]$ has a constant value in the two-valued ivasmode $V = \{T, F\}$ for r is in $\{1, 2, \ldots, 2^{2^n}$: n is in $P\}$ and i is in $\{0, 1, 2, 3, \ldots, 2^n - 1\}$. It is true that the first 2^{n-1} states in the Table 1 are realizable by $ux[n-1] \ldots ux[2]ux[1].\ ((F@sG[n-1])G[n-1])$ and the next 2^{n-1} states are realizable by $ux[n-1] \ldots ux[2]ux[1].\ ((T@sG[n-1])G[n-1])$ implies that $ux[n-1] \ldots ux[2]ux[1].\ ((x[n]@sG[n-1])G[n-1])$ realizes all states of the truth table appearing in Table 1 and the Fundamental Theorems of two-valued CITALOG closure automatically self minimizes them. Hence for all n in P a two-valued CITALOG closure $\%tC$ with a two-valued CITALOG operator closure $UC[n]$ and a two-valued CITALOG instruction closure $IC[n]$ realizes all two-valued n-ary logic operators and the Fundamental Theorems of two-valued CITALOG closure automatically self minimizes them.

Definition 2

(a) $G[n] = ((x[n]@sG[n-1])G[n-1])$ is called a two-valued n-ary packed TASIM body.

(b) The rules of expansion/construction of a two-valued CITALOG closure $\%tC$ in Theorem 3 are termed as the self minimizing rules of the two-valued CITALOG closure.

TABLE 1. The truth table for the rth operator in nUC

$x[n]$	$x[2]x[1]$	$nU[r]$	
F	$F\ F$	$c[0]$	
F	$F\ T$	$c[1]$	is realizable by $((F@sG[n-1])G[n-1])$
F	$T\ T$	$c[2^{n-1}-1]$	
T	$F\ F$	$c[2^{n-1}]$	is realizable by $((T@sG[n-1])G[n-1])$
T	$T\ T$	$c[2^n-1]$	

Corollary 1

A TASIM closure tC which covers a two-valued CITALOG closure $\%tC$ realizes and automatically minimizes any arbitrarily given two-valued logic function in the Logic Design.

Corollary 2

If x and y are two two-valued variables on the ivasmode $V = \{T, F\}$ in a two-valued

CITALOG closure $\%tC$, and x' and y' are their complements on V then $\%tC$ has the following partitions of semantic equality for logical software minimizing instructions on V:

$$
\begin{aligned}
&(1)\ F &&= ((x@F)F) &&= ((x@F)x) &&= ((x@x')F\\
& &&= ((x@x')x) &&= ((x'@F)F) &&= ((x'@F)x')\\
& &&= ((x'@x)F) &&= ((x'@x)x'),\\
&(2)\ T &&= ((x@T)T) &&= ((x@T)x') &&= ((x@x)T\\
& &&= ((x@x')x) &&= ((x'@T)T) &&= ((x'@T)x)\\
& &&= ((x'@)T) &&= ((x'@x')x),\\
&(3)\ x &&= ((x@T)F) &&= ((x@T)x) &&= ((x@x)F\\
& &&= ((x@x)x) &&= ((x'@F)T) &&= ((x'@F)x)\\
& &&= ((x'@x)T) &&= ((x'@x)x),\\
&(4)\ x' &&= ((x@F)T) &&= ((x@F)x') &&= ((x@x')T\\
& &&= ((x@x')x') &&= ((x'@T)F) &&= ((x'@T)x')\\
& &&= ((x@x')F) &&= ((x'@x')x'),\\
&(5)\ y &&= ((x@y) &&= ((x'@y)y),\\
&(6)\ y' &&= ((x@y')y') &&= ((x'@y')y').
\end{aligned}
$$

Applications

Example 1

If,

$n = 1$

and

(a) $V = \{\, T: = ux[1]ux[2].x[1],\ F: = ux[1]ux[2].\ x[2]\,\}$ is a 2-valued ivasmode of the 2-valued CITALOG closure,

(b) $x[1]$ is a 2-valued CITALOG variable and $G[1]$ is a unary TASIM instruction,

then,

one may determine $G[1] = ((x[1]@c[1])\ c[0])$, for $n = 1$;

else, if

$x[1], x[2], \ldots, x[n]$ are 2-valued CITALOG variables and $G[n]$ is an 2-valued n-ary packed TASIM body.

Then

one may determine $G[n] = ((x[n]@sG[n-1])\ G[n-1])$, for $n \geq 2$,

where

$sG[n-1] = c[i]Sc[ii]@G[n-1],\ ii = i + 2^{n-1}$.

Hence we have :

(i) $G[1] = ((x[1]@c[1])c[0])$, for $n = 1$,

(ii) $G[2] = ((x[1]@sG[1])G[1])$, for $n = 2$, where

$$sG[1] = c[i]Sc[ii]@((x[1]@c[1])c[0]) = ((x[1]@c[3])c[2]), \text{ for } ii = i + 2^1 = i + 2,$$

implies that

$$G[2] = ((x[2]((x[1]@c[3])c[2]))((x[1]@c[1])c[0])),$$

and

$$U[2] = ux[2]ux[1].G[2]$$
$$= ux[2]ux[1].((x[2]((x[1]@c[3])c[2]))((x[1]@c[1])c[0]))$$ is a 2-valued 2-ary CITALOG operator.

This operator realizes all binary and 2-valued logic operator and functions in the Logic Design.

Example 2

By the same derivation method in Example 1 :

(i) $U[3] = ux[3]ux[2]ux[1].G[3]$

$= ux[3]ux[2]ux[1].(\,(x[3]@((x[2]@((x[1]@c[7])\quad c[6])))$
12 34 56 6 543

$((x[1]@c[5])c[4])))((x[2]@((x[1]@c[3])c[2]))$
45 5 43223 45 5 43

$((x[1]@c[1])c[0])))$
34 4 321

is a 2-valued 3-ary CITALOG operator. It realizes all three input and one output logic gates or logic networks in the Logic Design.

(ii) $U[4] = ux[4]ux[3]ux[2]ux[1].G[4]$

where

$G[4] = ((x[4]@((x[3]@((x[2]@((x[1]@c[15])c[14]))((x[1]@c[13])c[12])))$
12 34 56 78 8 7667 7 654

$((x[2]@((x[1]c[11])c[10]))((x[1]@c[9])c[8]))))((x[3]@((x[2]@$
45 67 7 6556 6 543223 45

$((x[1]@c[7])c[6]))((x[1]c[5])c[4])))((x[2]@((x[1]@c[3])c[2]))$
67 7 6556 6 54334 56 6 54

$((x[1]@c[1])c[0]))))$
45 5 4321

is a two-valued 4-ary CITALOG operator. It realizes all four input and one output logic gates or logic networks in the Logic Design.

Example 3

Let $n = 3$, a 2-valued CITALOG closure realizes all 2-valued 3-ary logic operators and functions in the Logic Design since we can write a TASIM program for realizing and automatically minimizing any 3-ary Boolean function in the Logic Design. For example, let us realize $f = f(x[3], x[2], x[1]) = \Pi M(0, 2, 4, 6, 7) = M0\, M2\, M4\, M6\, M7$, where 'Π' is representing cummulative '*AND*' operator (Table 2) :

Step 1

$$f = ux[3]ux[2]ux[1].\underset{1}{(}\underset{2}{(}x[3]@\underset{3}{(}\underset{4}{(}x[2]@\underset{5}{(}\underset{6}{(}x[1]@c[7]\underset{6}{)}c[6]\underset{5}{)}\underset{4}{)}\underset{4}{(}\underset{5}{(}x[1]@c[5]\underset{5}{)}c[4]\underset{4}{)}\underset{3}{)}\underset{2}{)}$$

$$\underset{2}{(}\underset{3}{(}x[2]@\underset{4}{(}\underset{5}{(}x[1]@c[3]\underset{5}{)}c[2]\underset{4}{)}\underset{3}{)}\underset{3}{(}\underset{4}{(}x[1]@c[1]\underset{4}{)}c[0]\underset{3}{)}\underset{2}{)}\underset{1}{)}$$

$$= ux[3]ux[2]ux[1].\underset{1}{(}\underset{2}{(}x[3]@\underset{3}{(}\underset{4}{(}x[2]@\underset{5}{(}\underset{6}{(}x[1]@0\underset{6}{)}0\underset{5}{)}\underset{4}{)}\underset{4}{(}\underset{5}{(}x[1]@1\underset{5}{)}0\underset{4}{)}\underset{3}{)}\underset{2}{)}$$

$$\underset{2}{(}\underset{3}{(}x[2]@\underset{4}{(}\underset{5}{(}x[1]@1\underset{5}{)}0\underset{4}{)}\underset{3}{)}\underset{3}{(}\underset{4}{(}x[1]@1\underset{4}{)}0\underset{3}{)}\underset{2}{)}\underset{1}{)}$$

TABLE 2. The truth table for $f = \Pi\, M(0, 2, 4, 6, 7)$.

dN	$x[3]x[2]x[1]$	$f(x[3], x[2], x[1])$	
0	0 0 0	0	$c[0]$
	0 0 1	1	$c[1]$
2	0 1 0	0	$c[2]$
3	0 1 1	1	$c[3]$
4	1 0 0	0	$c[4]$
5	1 0 1	1	$c[5]$
6	1 1 0	0	$c[6]$
7	1 1 1	0	$c[7]$

Step 2

If two-valued CITALOG closure automatically minimizes it then we obtain

$$f = ux[3]ux[2]ux[1].\underset{1}{(}\underset{2}{(}x[3]@\underset{3}{(}\underset{4}{(}x[2]@0\underset{4}{)}x[1]\underset{3}{)}\underset{2}{)}\underset{2}{(}\underset{3}{(}x[2]@x[1]\underset{3}{)}x[1]\underset{2}{)}\underset{1}{)}$$

$$= ux[3]ux[2]ux[1].\underset{1}{(}\underset{2}{(}x[3]@\underset{3}{(}\underset{4}{(}x[2]@0\underset{4}{)}x[1]\underset{3}{)}\ \underset{2}{)}x[1]\underset{1}{)}$$

Step 3

If we process f for $x[3] = 0$ and $x[2] = x[1] = 1$, we obtain :

$$\underset{1}{(}\underset{2}{(}\underset{3}{(}f@0\underset{3}{)}1\underset{2}{)}1\underset{1}{)} = \underset{1}{(}\underset{2}{(}\underset{3}{(}ux[3]ux[2]ux[1].\ \overset{1}{\underset{4}{(}}\overset{2}{\underset{5}{(}}x[3]@\overset{3}{\underset{6}{(}}\overset{4}{\underset{7}{(}}x[2]@0\overset{4}{\underset{7}{)}}x[1]\overset{3}{\underset{6}{)}}\overset{2}{\underset{5}{)}}x[1]\overset{1}{\underset{4}{)}}\,0\underset{3}{)}1\underset{2}{)}1\underset{1}{)}$$

```
= ((0((1@0)1))1)
  12 34    4 32 1

= ((0@0)1)
  12    2 1

= 1
```

Example 4

Let

```
G = ((x[3]@((x[2]@((x[1]@((x[0]@a[15])a[14]))((x[0]@a[13])a[12])))
    12      34      56      78             8      7667            7      654

    ((x[1]@((x[0]@a[11])a[10]))((x[0]@a[9])a [8]))))
    45      67             7      6556           6     5432

    ((x[2]@((x[1]@((x[0]@a[7])a[6]))((x[0]@a[5])a[4])))((x[1]@
    23      45      67            7     6556           6     54334

    ((x[0]@a[3])a[2]))((x[0]@a[1])a[0])))),
    56            6     5445           5   4321
```

then

$$4U = ux[3]ux[2]ux[1]ux[0].G$$

realizes all 2-valued 4-ary logic functions in dual space for the CITALOG variables $x[0]$, $x[1]$, $x[2]$, $x[4]$ and two-valued CITALOG constants $a[0]$, $a[1]$, ..., $a[15]$. For example, if one wishes to realize

$$f' = \Pi M(0, 1, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

in dual space :

```
f = ux[3]ux[2]ux[1]ux[0].((x[3]@((x[2]@((x[1]@
                           12      34      56

    |← x[ 0 ]→| |← x[ 0 ]→|
    ((x[0]@1)0))((x[0]@1)0)))
    78         8 7667       7 654

             |←  1    →| |←  1    →|
    ((x[1]@((x[0]@1)1))((x[0]@1)1))))
    45      67        7 6556       6 5432

                     |← x[ 0 ]→| |←  x[ 0 ] →|
    ((x[2]((x[1]@((x[0]@1)0))((x[0]@1) 0) ))
    23     45      67        7 6556        6 5 43

              |← 1    →| |←  1    →|
    ((x[1]@((x[0]@1)1))((x[0]@1)1))))
    34      56        6 5445       5 4321
```

$$\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad |\leftarrow \quad x[0] \quad \rightarrow| \\
f = ux[3]ux[2]ux[1]ux[0].((x[3]@((x[2]@((x[1]@x[0])x[0])) \\
\qquad\qquad\qquad\qquad\qquad\quad 12 \qquad\quad 34 \qquad\quad 56 \qquad\qquad 6 \quad 54
\end{array}$$

$$\begin{array}{l}
|\leftarrow 1 \quad \rightarrow| \\
((x[1]@1)1))) \\
45 \qquad\quad 5 \; 432
\end{array}$$

$$\begin{array}{l}
\qquad\quad |\leftarrow \quad x[0] \quad \rightarrow| \; |\leftarrow 1 \quad \rightarrow| \\
((x[2]@((x[1]@x[0)x[0])) \; ((x[1]@1)1))) \\
23 \qquad 45 \qquad\qquad 5 \quad 43\; 34 \qquad\quad 4 \; 321
\end{array}$$

$$\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad |\leftarrow \; A \quad \rightarrow| \; |\leftarrow \; A \quad \rightarrow| \\
f = ux[3] \; ux[2] \; ux[1] \; ux[0].((x[3]@ \;\; ((x[2]@x[0])1)) \; ((x[2]@x[0])1)) \\
\qquad\qquad\qquad\qquad\qquad\quad 12 \qquad\quad 34 \qquad\qquad 4 \; 32 \; 23 \qquad\qquad 3 \; 21
\end{array}$$

$f = ux[3]ux[2]ux[1]\, ux[0].((x[2]@ux[0])1)$, in dual space.

TABLE 3*a*. The truth table of *f*.

		f	
0	0000		*a*[0]
1	0001		*a*[1]
2	0010	1	*a*[2]
3	0011	1	*a*[3]
4	0100	0	*a*[4]
5	0101	1	*a*[5]
6	0110	0	*a*[6]
7	0111	1	*a*[7]
8	1000	1	*a*[8]
9	1001	1	*a*[9]
10	1010	1	*a*[10]
11	1011	1	*a*[11]
12	1100	0	*a*[12]
13	1101	1	*a*[13]
14	1110	0	*a*[14]
15	1111	1	*a*[15]

Now we may test this self minimizing function by the truth table of *f* given in Table 3*a* and compare it by the Boolean function that one can obtain from the *K*-map appearing in Table 3*b*.

$$\begin{aligned} f = ux[3]ux[2]ux[1]ux[0].((x[2]@x[0])1) &\rightarrow f = x[2]x[0]' \\ &\rightarrow f' = x[2]' + x[0]. \end{aligned}$$

Conclusion

In this paper, we have developed a new technique for two-valued logic software minimizing information processing algorithms as TASIM programs in the two-valued CITALOG closure for realizing two-valued logic operators and functions of any finite number of operands and any finite number of variables. It has a power of self minimizing automatically two-valued logic functions with any finite number of variables in any two-valued logic. This may help to create a new technology in the Logic Design and also may produce software minimization techniques for realizing algorithms and data structures. In this line, the author is studying on (a) a two-valued minimizing CITALOG virtual machine as a software structure for minimizing any finite two-valued logic function, (b) a minimizing TASIM virtual machine for minimizing software structures in TASIM.

The self software minimizing power that we have coded into the instructional information processing structures of the two-valued CITALOG closure is a milestone in the Logic Design to point out the way of producing a new technology for optimal hardware or optimal software system development.

TABLE 3*b*. K-map of *f*.

$x[1]x[0]$ \ $x[3]x[2]$	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	1	1
10	1	0	0	1

K-map

References

[1] **Denning, P.J., Dennis, J.B.** and **Qualitz, J.E.,** *Machines, Languages, and Computation,* Prentice Hall, Inc., Engelwood Cliffs, New Jersey (1978).

[2] **Ünlü, F.,** *Theoretical lambda-TASIM,* Atatürk University, Erzurum, Pub. No. **472,** 40 p. (1976).

[3] **Roth, C.H.,** *Fundamentals of Logic Design,* 3*rd* ed., West Publishing Company, New York (1985).

[4] **Ünlü, F.,** ü-TASIM, *6th National Congress of Operation Research,* 25-27 June, Hacettepe University, Ankara, unpublished (1980).

[5] **Mirasyedioğlu, S.,** *Operator Classification in lambda-TASIM by a Logical Evaluation Process,* Ph.D. Thesis, Ataturk University, Erzurum (1979).

[6] **Albayrak, L.,** *Derivation of Some Algebraic Structures in a lambda-Culture,* Ph.D. Thesis, Department of Mathematics, Ege University, Izmir (1982).

[7] **Ünlü, F.,** A TASIM Logic Realization of Boolean Algebra, *DIRASAT: A Research Journal,* the University of Jordan, **XIII(7):** 67-76 (1986).

[8] **Ünlü, F.,** TASIM Logic Realizations in the Logic Design, *DIRASAT: A Learned Research Journal,* the University of Jordan, **XIV(12):** 61-80 (1987).

[9] **Pratt, T.W.,** *Programming Languages: Design and Implementation,* (Second Edition), Prentice Hall International, Inc., Engelwood Cliffs, New Jersey (1984).

Appendix
The Truth Tables for the Unary and Binary Logic Operators

This appendix is for introducing the truth tables of unary and binary operators in the Logic Design. For this reason Table 1 contains the four possible two-valued operators and Table 2 contains the sixteen possible two-valued binary operators. Where we use $x[i] = xi$, $c[k] = ck$, $1U[j] = 1Uj$ and $2U[k] = 2Uk$.

TABLE 1. The truth table for the two-valued unary logic operators in the Logic Design.

*x*1	1*U*1	1*U*2	1*U*3	1*U*4	*c*
F	*F*	*T*	*F*	*T*	*c*0
T	*F*	*F*	*T*	*T*	*c*1

TABLE 2. The truth table for the two-valued binary logic operators in the Logic Design.

*x*2*x*1	2*U*1	2*U*2	2*U*3	2*U*4	2*U*5	2*U*6	2*U*7	2*U*8	2*U*9	2*UA*	2*UB*	2*UC*	2*UD*	2*UE*	2*UF*	2*UG*	*c*
F F	*F*	*T*	*F*	*T*	*F*	*T*	*F*	*T*	*F*	*T*	*F*	*T*	*F*	*T*	*F*	*T*	*c*0
F T	*F*	*F*	*T*	*T*	*F*	*F*	*T*	*T*	*F*	*F*	*T*	*T*	*F*	*F*	*T*	*T*	*c*1
T F	*F*	*F*	*F*	*F*	*T*	*T*	*T*	*T*	*F*	*F*	*F*	*F*	*T*	*T*	*T*	*T*	*c*2
T T	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*c*3

1*U*1 :	Unary contradiction,	2*UB*	Binary	identity *x*1,
1*U*2 :	„ negation,	2*UC*	„	conditional,
1*U*3 :	„ identity,	2*UD*	„	identity of *x*2,
1*U*4 :	„ tautology,	2*UE*	„	converse,
2*U*1 :	Binary contradiction,	2*UF*	„	inclusive *OR*,
2*U*2 :	„ *NOR*,	2*UG*	„	tautology,
2*U*3 :	„ negation of converse,	*A*	10,	
2*U*4 :	„ negation, of *x*2,	*B*	11,	
2*U*5 :	„ negation of conditional,	*C*	12,	
2*U*6 :	„ negation of *x*1,	*D*	13,	
2*U*7 :	„ exclusive *OR*,	*E*	14,	
2*U*8 :	„ *NAND*,	*F*	15,	
2*U*9 :	„ *AND*,	*G*		
2*U*A	„ biconditional,			

سيـــــتالوج
الانغلاق المتكامل المضغوط للمنطق تاسيم

فوزي محمد أونلو .

قسم علم الحاسبات - كلية العلوم - جامعة الملك عبد العزيز

جـــدة - المملكة العربية السعودية

يعرض هذا البحث تطويرًا للانغلاق المتكامل المضغوط للمنطق التاسيمي (سيتالوج) ثنائي القيمة لتقليص البرامج . يهدف هذا التطوير إلى إيجاد مشغلات منطقية ودوال ثنائية القيمة تتعامل مع أي عدد من المعاملات أو المتحولات ، وذلك حسب قواعد الإنشاء والتشغيل المحـددة في تاسيم . بذلـك يُصبـح من الممكن إيجاد التصاميم البراجية ذاتية التقليص للهياكل الآلية في التصميم المنطقي .