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 Mahallesı, 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\(^\text{[7]}\). 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\(^\text{[8]}\). The notion of high level programming languages can be found in Ref.\(^\text{[9]}\).

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

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


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


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

$L1$: A two-valued CITALOG closure has two distinct elements $F$ and $T$ in $%aT$ implies that $F$ and $T$ are in $aT$.


$L3$: Let $x[0], x[1], \ldots, x[n]$ be $n + 1$ TASIM variables taken from $%aLC[j]$ and $c[0], c[1], \ldots, c[m]$, $m = 2^n - 1$ be names for constant data objects also taken from $%aLC[j]$ assuming values on $V$.

(a) $(B1)$ 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) $UC[n] = \{ U[n] : n \in P \}$ is called a two-valued CITALOG operator closure in $%tC$.

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

$L4$: Let $y[1], y[2], \ldots, y[n]$ 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 $V \subseteq %tC$, for $n \in P$.

(a) $(I1)$ If $I[1] = (U[1]@y[1]) = (ux[1].G[1]@y[1]) = x[1]Sy[1]@G[1]$ is called a unary instruction in the two-valued CITALOG closure $%tC$ for $n = 1$.


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

(c) $IC[n]$ 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:

(a) \( ([T@R]Q) = ([ux[1]ux[2]\cdot x[1]@R]Q) = x[1]SR@x[2]SQ@x[1] = R \),

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. \( ([x'@T]F) = x' \),
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:

\begin{align*}
(1) \quad (i) & \text{ For } x = F, \quad ((x@F)F) = ((F@F)F) = F. \\
(ii) & \text{ For } x = T, \quad ((x@F)F) = ((T@F)F) = F.
\end{align*}

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

\begin{align*}
(2) \quad (i) & \text{ For } x = F, \quad ((x@F)T) = ((F@F)T) = T = x' \\
(ii) & \text{ For } x = T, \quad ((x@F)T) = ((T@F)T) = F = x'.
\end{align*}

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

\begin{align*}
(3) \quad (i) & \text{ For } x = F, \quad ((x@F)x) = ((F@F)F) = F. \\
(ii) & \text{ For } x = T, \quad ((x@F)x) = ((T@F)F) = F.
\end{align*}

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

\begin{align*}
(32) \quad (i) & \text{ For } x = F, \quad ((x'@x')x') = ((T@T)T) = T = x'. \\
(ii) & \text{ For } x = T, \quad ((x'@x')x') = ((F@F)F) = F = x'.
\end{align*}

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

\begin{align*}
(33) \quad (i) & \text{ For } x = F, \quad y = F, \quad ((x@y)y) = ((F@F)F) = F = y. \\
(ii) & \text{ For } x = F, \quad y = T, \quad ((x@y)y) = ((F@T)T) = T = y. \\
(iii) & \text{ For } x = T, \quad y = F, \quad ((x@y)y) = ((T@F)F) = F = y. \\
(iv) & \text{ For } x = T, \quad y = T, \quad ((x@y)y) = ((T@T)T) = T = y.
\end{align*}

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 $n = 1$, \(U = ux[1].G[1] = ux[1].((x[1]@c[1])c[0])\). 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 $c[1] = T$, and
   - (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 $n = 2$,


$$= ux[2]ux[1].((x[2]@sG[1])G[1])$$

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

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:


on the ivasmode $V$.

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


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

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


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


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


(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) \(((2U[3]@T)T) = ((T@F)T) = F.\)


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

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) \(((2U[5]@F)T) = ((F@F)T) = F,\)
(iii) \(((2U[5]@T)F) = ((T@F)F) = T,\)
(iv) \(((2U[5]@T)T) = ((T@F)T) = F.\)

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) \(((2U[6]@T)F) = ((T@F)T) = T,\)
(iv) \(((2U[6]@T)T) = ((T@F)F) = F.\)

\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (A)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (B)}\]
\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (C)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (D)}\]

is obtained for \(c_0 = c_2 = T, c_1 = F\) and
\[L = [1] \circ F = [2] \circ F = [1] \circ F = [0] \circ F = [0]\]


\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (A)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (B)}\]
\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (C)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (D)}\]

is obtained for \(c_0 = c_2 = T, c_1 = F\) and
\[L = [1] \circ F = [2] \circ F = [1] \circ F = [0] \circ F = [0]\]


\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (A)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (B)}\]
\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (C)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (D)}\]

is obtained for \(c_0 = c_2 = T, c_1 = F\) and
\[L = [1] \circ F = [2] \circ F = [1] \circ F = [0] \circ F = [0]\]


\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (A)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (B)}\]
\[L = (L \odot L) = (L \odot [L \odot 2]) \text{ (C)}\]
\[F = (L \odot L) = (L \odot [L \odot 2]) \text{ (D)}\]

is obtained for \(c_0 = c_2 = T, c_1 = F\) and
\[L = [1] \circ F = [2] \circ F = [1] \circ F = [0] \circ F = [0]\]

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

(14) \[ U[14] = u[x[2]]u[x[1]].((x[2]@((x[1]@T)T)))(x[1]@F)F) \]

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

(15) \[ U[15] = u[x[2]]u[x[1]].((x[2]@((x[1]@T)T)))(x[1]@T)F) \]

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

(16) \[ U[16] = u[x[2]]u[x[1]].((x[2]@((x[1]@T)T)))(x[1]@T)F) \]

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

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:

(1) Assume for all \( c[i] \) in \( G[n-1] \), we define \( sG[n-1] = c[i]Sc[i]@G[n-1] \),

\[ ii = i + 2^{n-1} \], and \( G[n] = ((x[n]sG[n-1]) G[n-1]) \) that \( U[n-1] = u[x[n-1]] \ldots u[x[2]]u[x[1]].G[n-1] \) in \( UC[n] \) realizes all two-valued \( (n-1) \)-ary operators and the Fundamental Theorems of two-valued CITALOG closure automatically self minimizes them is true for \( n \geq 2 \). Then, by induction on \( n \) in \( P \), \( u[x[n]] \ldots u[x[2]]u[x[1]].G[n] \) in \( UC[n] \) realizes all two-valued \( n \)-ary operators for \( G = ((x[n]sG[n-1])G \)
and the Fundamental Theorems of two-valued CITALOG closure automatically minimizes them is also true. Because the $r$th 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 $\{1, 2, ..., 2^n\}$. $n$ is in $P$ and $i$ is in $\{0, 1, 2, 3, ..., 2^n - 1\}$. It is true that the first $2^n - 1$ states in the Table 1 are realizable by $ux[n - 1] ... ux[2]ux[1]$. ($((F@sG[n - 1])G[n - 1])$ and the next $2^n - 1$ states are realizable by $ux[n - 1] ... ux[2]ux[1]$. ($((T@sG[n - 1])G[n - 1])$ implies that $ux[n - 1] ... ux[2]ux[1]$. ($((x[n]@sG[n - 1])G[n - 1])$ 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 $r$th operator in $nUC$**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$F F$</td>
<td>$c[0]$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F T$</td>
<td>$c[1]$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T T$</td>
<td>$c[2^n - 1]$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F F$</td>
<td>$c[2^n - 1]$</td>
</tr>
<tr>
<td>$T$</td>
<td>$T T$</td>
<td>$c[2^n - 1]$</td>
</tr>
</tbody>
</table>

**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
CITATALOG 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:

(1) \( F = ((x@F)x) = ((x@x')F = ((x@x)x) = ((x@x')F = ((x@x)x) = ((x@x')F.\)

(2) \( T = ((x@T)x) = ((x@x)x) = ((x@T)x) = ((x@T)x) = ((x@T)x) = ((x@T)x).\)

(3) \( x = ((x@x)x) = ((x@x)x) = ((x@x)x) = ((x@x)x) = ((x@x)x).\)

(4) \( x' = ((x@x)x) = ((x@x)x) = ((x@x)x) = ((x@x)x).\)

(5) \( y = ((x@y)x) = ((x@y)x).\)

(6) \( y' = ((x@y)x) = ((x@y)x).\)

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 CITATALOG closure,

(b) \( x[1] \) is a 2-valued CITATALOG 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 \( n \) are 2-valued CITATALOG 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

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

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

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

implies that

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

and

\[ = ux[2]ux[1].((x[2])((x[1]@c[3])c[2]))((x[1]@c[1])c[0])) \]

is a 2-valued 2-ary CIT ALOG 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:

\[ ((x[1]@c[4]))((x[2]@((x[1]@c[3])c[2]))) \]
\[ ((x[1]@c[1])c[0])) \]

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


where

\[ G[4] = ((x[4]@((x[3]@((x[2]@((x[1]@c[15])c[14])))((x[1]@c[13])c[12]))) \]
\[ ((x[2]@((x[1]@c[11])c[10]))((x[1]@c[9])c[8])))((x[3]@((x[2]@ \]
\[ ((x[1]@c[7])c[6]))((x[1]@c[5])c[4])))((x[2]@((x[1]@c[3])c[2]))) \]
\[ ((x[1]@c[1])c[0]))) \]

is a two-valued 4-ary CIT ALOG 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 ‘\( \Pi \)’ is representing cummulative ‘\( \text{AND} \)’ operator (Table 2):

\[
\text{Step 1}
\]
\[
f = \{x[3]\}u\{x[2]\}u\{x[1]\}.(\{x[3]\}u\{x[2]\}u\{x[1]\}c[7])c[6])c[5])c[4])
\]
\[
\quad = \{x[3]\}u\{x[2]\}u\{x[1]\}.(\{x[3]\}u\{x[2]\}u\{x[1]\}c[0])c[4])
\]

\[
\text{Table 2. The truth table for } f = \Pi M(0, 2, 4, 6, 7).
\]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 c[0]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 c[1]</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 c[2]</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 c[3]</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 c[4]</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 c[5]</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 c[6]</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 c[7]</td>
</tr>
</tbody>
</table>

\[
\text{Step 2}
\]
If two-valued CITALOG closure automatically minimizes it then we obtain
\[
f = \{x[3]\}u\{x[2]\}u\{x[1]\}.(\{x[3]\}u\{x[2]\}u\{x[1]\}c[0]c[2])c[4])
\]
\[
\quad = \{x[3]\}u\{x[2]\}u\{x[1]\}.(\{x[3]\}u\{x[2]\}u\{x[1]\}c[0]c[2])c[4])
\]

\[
\text{Step 3}
\]
If we process \( f \) for \( x[3] = 0 \) and \( x[2] = x[1] = 1 \), we obtain:
\[
(((f@0)@1)@1) = (((\{x[3]\}u\{x[2]\}u\{x[1]\}.(\{x[3]\}u\{x[2]\}u\{x[1]\}c[0]c[2])c[4])c[2])c[4])c[2])c[4])
\]
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 8 7 6566 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])))),
5 6 6 5446 5 4321

then

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 \( a[0] \), \( a[1] \), \( \ldots \), \( a[15] \). 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 :

12 34 56
\leftarrow x[0 ] \rightarrow) | \leftarrow x[0 ] \rightarrow)
((x[0]@[1]0))((x[0]@[1]0)))
78 8 7666 7 654
\leftarrow 1 \rightarrow) | \leftarrow 1 \rightarrow)
((x[1]@((x[0]@[1]1)))((x[0]@[1]1))))))
45 67 7 6556 6 5432
\leftarrow x[0 ] \rightarrow) | \leftarrow x[0 ] \rightarrow)
((x[2]@((x[1]@((x[0]@[1]0))((x[0]@[1]0)))))
23 45 67 7 6556 6 5432
\leftarrow 1 \rightarrow) | \leftarrow 1 \rightarrow)
((x[1]@((x[0]@[1])1))((x[0]@[1])1)))
34 56 6 5446 5 4321
\[ f = \text{ux}[3]\text{ux}[2]\text{ux}[1]\text{ux}[0].((\text{ux}[3]@((\text{x}[2]@((\text{x}[1]@\text{x}[0])\text{x}[0])))) \]

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

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

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

### Table 3a. The truth table of \( f \).

<table>
<thead>
<tr>
<th>( \cdots \cdots \cdots \cdots \cdot f )</th>
<th>( \cdots \cdots \cdots \cdots \cdot a[0] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0000</td>
<td>( a[0] )</td>
</tr>
<tr>
<td>1 0001</td>
<td>( a[1] )</td>
</tr>
<tr>
<td>2 0010</td>
<td>1 ( a[2] )</td>
</tr>
<tr>
<td>3 0011</td>
<td>1 ( a[3] )</td>
</tr>
<tr>
<td>4 0100</td>
<td>0 ( a[4] )</td>
</tr>
<tr>
<td>5 0101</td>
<td>1 ( a[5] )</td>
</tr>
<tr>
<td>6 0110</td>
<td>0 ( a[6] )</td>
</tr>
<tr>
<td>7 0011</td>
<td>1 ( a[7] )</td>
</tr>
<tr>
<td>8 1000</td>
<td>1 ( a[8] )</td>
</tr>
<tr>
<td>9 1001</td>
<td>1 ( a[9] )</td>
</tr>
<tr>
<td>10 1010</td>
<td>1 ( a[10] )</td>
</tr>
<tr>
<td>11 1011</td>
<td>1 ( a[11] )</td>
</tr>
<tr>
<td>12 1100</td>
<td>0 ( a[12] )</td>
</tr>
<tr>
<td>13 1101</td>
<td>1 ( a[13] )</td>
</tr>
<tr>
<td>14 1110</td>
<td>0 ( a[14] )</td>
</tr>
<tr>
<td>15 1111</td>
<td>1 ( a[15] )</td>
</tr>
</tbody>
</table>

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 CATALOG 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 CIT ALOG 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 CIT ALOG 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>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

K-map

References

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 \( x[i] = x_i \), \( c[k] = c_k \), \( 1U[j] = 1U_j \) and \( 2U[k] = 2U_k \).

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

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( 1U_1 )</th>
<th>( 1U_2 )</th>
<th>( 1U_3 )</th>
<th>( 1U_4 )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>( F )</td>
<td>( T )</td>
<td>( F )</td>
<td>( T )</td>
<td>( c_0 )</td>
</tr>
<tr>
<td>( T )</td>
<td>( F )</td>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
<td>( c_1 )</td>
</tr>
</tbody>
</table>

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

| \( x_2 \times x_1 \) | \( 2U_1 \) | \( 2U_2 \) | \( 2U_3 \) | \( 2U_4 \) | \( 2U_5 \) | \( 2U_6 \) | \( 2U_7 \) | \( 2U_8 \) | \( 2U_9 \) | \( 2U_{10} \) | \( 2U_{11} \) | \( 2U_{12} \) | \( 2U_{13} \) | \( 2U_{14} \) | \( 2U_{15} \) | \( 2U_{16} \) | \( c \) |
|---------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|------|
| \( F \times F \)    | \( F \)  | \( F \)  | \( F \)  | \( F \)  | \( T \)  | \( F \)  | \( T \)  | \( F \)  | \( F \)  | \( F \)  | \( T \)  | \( F \)  | \( T \)  | \( F \)  | \( F \)  | \( F \)  | \( F \)  | \( c_0 \) |
| \( F \times T \)    | \( F \)  | \( T \)  | \( F \)  | \( T \)  | \( T \)  | \( T \)  | \( F \)  | \( T \)  | \( F \)  | \( T \)  | \( T \)  | \( T \)  | \( F \)  | \( T \)  | \( F \)  | \( T \)  | \( T \)  | \( c_1 \) |
| \( T \times T \)    | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( T \)  | \( c_2 \) |

\[
\begin{align*}
1U_1 & : \text{Unary contradiction,} \\
1U_2 & : \text{" negation,} \\
1U_3 & : \text{" identity,} \\
1U_4 & : \text{" tautology,} \\
2U_1 & : \text{Binary contradiction,} \\
2U_2 & : \text{" NOR,} \\
2U_3 & : \text{" negation of converse,} \\
2U_4 & : \text{" negation of } x_2, \\
2U_5 & : \text{" negation of conditional,} \\
2U_6 & : \text{" negation of } x_1, \\
2U_7 & : \text{" exclusive OR,} \\
2U_8 & : \text{" NAND,} \\
2U_9 & : \text{" AND,} \\
2UA & : \text{" biconditional,}
\end{align*}
\]

\[
\begin{align*}
2UB & : \text{Binary identity } x_1, \\
2UC & : \text{" conditional,} \\
2UD & : \text{" identity of } x_2, \\
2UE & : \text{" converse,} \\
2UF & : \text{" inclusive OR,} \\
2UG & : \text{" tautology,}
\end{align*}
\]
سيتالوج
الانغلاق المتكامل المضغوط للمنطق تاسيم

فوزي محمد أولتو
قسم علم الحاسبات - كلية العلوم - جامعة الملك عبد العزيز
جدة - المملكة العربية السعودية

يعرض هذا البحث تطورًا للانغلاق المتكامل المضغوط للمنطق التاسيمي (سيتالوج) ثنائي القمية لتقييم البرامج. يهدف هذا التطور إلى إيجاد مشغولات منطقية ودوال ثنائية القمية تتعامل مع أي عدد من المعاملات أو التحولات، وذلك حسب قواعد الإنشاء والتشغيل المحددة في تاسيم. بذلك يصبح من الممكن إيجاد التصميم البرمجي ذاتية التقييم للمهياكل الآلية في التصميم المنطقي.