Algorithm For Optimal Implementation of Multiple-Valued Functions Using Switches

Dorin Sima, "Lucian Blaga" University, Bd. Victoriei No. 10, Sibiu, Romania; dorin.sima@ulbsibiu.ro
Rodica Baciu, "Lucian Blaga" University, Bd. Victoriei No. 10, Sibiu, Romania; rodica.baciu@ulbsibiu.ro

Abstract: In this paper is presented an automatic synthesis algorithm, developed by authors, used for optimal ipmlementation of multiple-valued logic functions(MVL functions) using *multiple-valued switches(MVL switches)*. The algorithm starts with the representation of the multivalued logic functions as *Multivalued Decision Diagrams* (MDD), from which it extracts the expressions corresponding to each switch F_i, represented by expression trees. Further, one identify the rules that applied to the nodes of the tree lead to lower cost of implementation (smaller number of CMOS transistors). When no rule can be applied any more, we have the minimal expression of implementation for each switch F_i.

Keywords: Multiple-Valued Logic, MVL, Decision Diagrams, MDD, MVL-Switches

INRODUCTION

The most stressing problems regarding binary VLSI are those referring to interconnection, both on-chip and between-chip. The accepted solution for these problems consists of the M*ultiple-Valued Logic* systems (MVL). Stepping down of interconnections is due to the growing of informational content in the digital signal, rated to the binary case. In the last twenty years literature we find remarkable achievements of MVL circuits in different technologies: I^2L, CCD, voltage-mode CMOS, current-mode CMOS, etc. In the same time, variants of the Post multivalued algebra have been searched, which are closer to hardware implementation, so called implementation-oriented algebras. Among the most known algebras are those developed by Allen and Givone, Vranesic algebra (for I^2L, CCD) and Jain, Bolton and Abd-El-Barr algebra (for current-mode CMOS).

Hassan proposes a new direction (Hassan 1996). The basic idea consists of the implementation of any MVL function by replacing MVL gates by *multiple-valued switches* (Fig. 1). Each *MVL switch* F_i is accomplished by series, parallel or cascade *interconnection of sub-switches*(Fig. 2). In (Hassan 1996) is shown a manual method for optimal implementation of a multivalued function, based on the enforcement of associativity, commutativity and distributivity of the three types of interconnection of subswitches. In this paper is presented an automatic synthesis algorithm, developed by authors. The algorithm starts with the representation of the multivalued function as *Multivalued Decision Diagrams* (MDD), from which it extracts the expressions corresponding to each switch F_i, represented by expression trees. Further, one identify the rules that applied to the nodes of the tree lead to lower cost of implementation (smaller number of CMOS transistors). When no rule can be applied any more, we have the minimal expression of implementation for each switch F_i

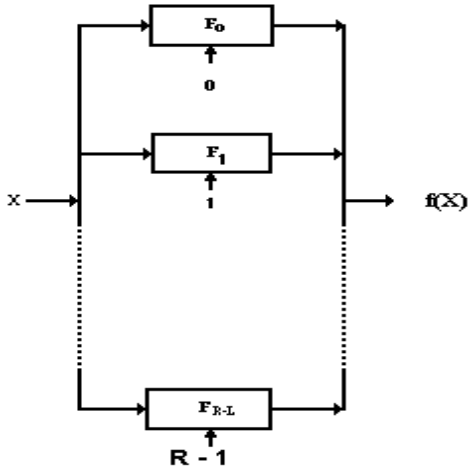

Fig. 1 MVL function implementation using MVL switches

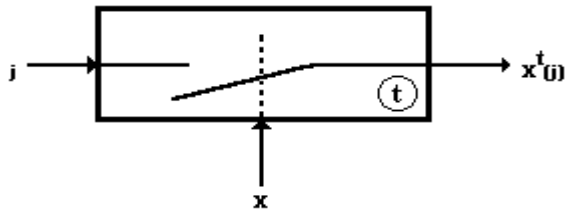

Fig. 2 Subswitches

BACKGROUND

Multiple-valued switches

Let F be a multiple-valued input, multiple-valued output function of n variables : $x_1, x_2,\ldots, x_n$.

$F : P_1 \times P_2 \times \ldots \times P_n \rightarrow Y$

Each variable, x_i, may take any p_i values from a finite set $P_i = \{0, 1, \ldots p_i - 1\}$. The output function F may take m values from the set $Y = \{0, 1, \ldots m - 1\}$.

Let T_i be a subset of P_i. The *Literal* of variable x_i is defined as the Boolean function:

$$x_i^{Ti} = \begin{cases} 0 \text{ if } xi \notin Ti \\ 1 \text{ if } xi \in Ti \end{cases}$$

The *Cofactor* of F with respect to a variable x_i taking a constant value *j* is:

$$F_{x_i^j} = F(x_1, \ldots x_{i-1}, \boldsymbol{j}, x_{i+1}, \ldots x_n),$$

the resulting function depend on n -1 variables
Other notations for cofactor: $F_{xi=j}$, $F^j{}_{xi}$

Note: If F does not depend on x_i, then $F_{xi=j} = F$.
The Shannon decomposition of a function F with respect to a variable x_i is:

$F = \sum_{j=0}^{pi-1} x_i^j \cdot F_{xi=j}$, where operations are max and min

Alternatively, the decomposition of a R-valued function f(X) could be expressed by:

$f(X) = \sum_{i=0}^{R-1} i \circ \vartheta_i(X)$, where $\vartheta_i(X)$ is the i-th MVL switch function and "o" is the transmission operator.

$$\vartheta_i(X) = \begin{cases} ON & \text{if } X \in T_i \\ OFF & \text{else} \end{cases}$$

Then, the i-th MVL switch is represented by F_i:

$F_i(X) = i \circ \vartheta_i(X)$

$$= \begin{cases} i & \text{if } X \in T_i \Leftrightarrow \vartheta_i(X) \text{ is ON} \\ OFF & \text{if } X \notin T_i \Rightarrow \vartheta_i(X) \text{ is OFF} \end{cases} \quad \textbf{(1)}$$

For example (from Hassan), the ternary function from table 1 could be represented:

x_1	x_2	$f(x_1, x_2)$
0	0	2
0	1	0
0	2	1
1	0	2
1	1	0
1	2	2
2	0	1
2	1	2
2	2	0

Table 1. Multivalued logic function

$f(x_1, x_2) = \sum_{i=0}^{2} i*\vartheta_i(x_1,x_2) = 0*\vartheta_0(x_1, x_2) + 1*\vartheta_1(x_1, x_2) + 2*\vartheta_2(x_1,x_2)$, where:

$\vartheta_0(x_1, x_2) =$

$$\begin{cases} ON & \text{if } (x_1, x_2) \in T_0 = \{(0,1),(1,1),(2,2)\} \\ 0 & \textit{else} \end{cases}$$

$\vartheta_1(x_1, x_2) =$

$$\begin{cases} ON & \text{if } (x_1, x_2) \in T_1 = \{(2,0),(0,2)\} \\ 0 & \text{else} \end{cases}$$

$\vartheta_2(x_1, x_2) =$

$$\begin{cases} ON & \text{if } (x_1, x_2) \in T_1 = \{(0,0),(1,0),(2,1),(1,2)\} \\ 0 & \text{else} \end{cases}$$

Each switch F_i is composed from more elementary elements called *subswitches* i.e. a switch with a single variable as an input control *(Fig. 2)*. The subswitch shown in Fig. 2 is controlled by a R-valued variable x, and "j" is the desired valued to be switched to output when x is in the threshold set "t":

$$x^t(j) = \begin{cases} j & \text{if } x \in t \\ OFF & \text{else} \end{cases} \quad \textbf{(2)}$$

$j \in S$ and $t \subseteq S$, $S = \{0, 1, 2, \ldots, R-1\}$

If Card(t)=1 than the corresponding subswitch is called "*unary subswitch*"

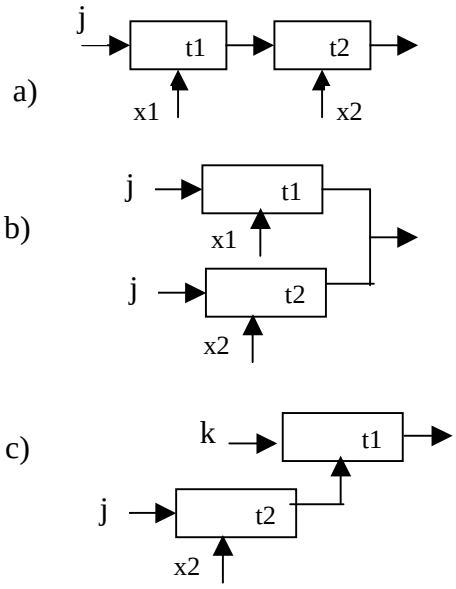

Fig. 3 Subswitches connection:
a)series b)parallel c)cascade

More complicated switches could be obtained by subswitches interconnection, as in figure 3. For more details see (Hassan 1996).
The series and parallel interconnections are commutative. These properties, together with distributive law are used (in the Hassan's paper) to give the optimal implementation. Optimal means a minimum number of CMOS transistors.

Multiple-valued Decisions Diagrams (MDD)

The state-of-the-art data structures used in the CAD systems for internal representations of logical functions are decision diagrams: Binary Decision Diagrams (BDD) for binary functions and, MDD for the multivalued functions. For more details about MDDs see (Kam 1995; Drechsler 1998).
The ternary function f(x1, x2) from the above example (Table 1) is represented by a MDD as in the Fig. 4. The edges outgoing each node are labeled from left to right with 0, 1 and 2. We can see that following all paths from root to leaves, each path corresponds to an individual row on truth table of the function.

OPTIMAL IMPLEMENTATION OF MVL FUNCTIONS

In this section we present the algorithm that determines the optimal expression of switches used to implement a given MVL function. We suppose that the function to be implemented is represented with MDD.

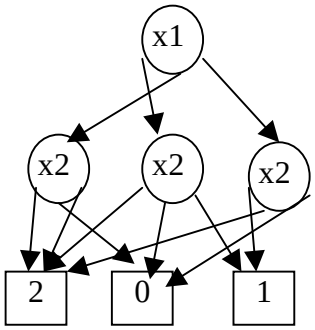

Fig. 4 The MDD representation of the function f(x1,x2) (see Table 1)

The algorithm has two main phases:

- Obtain the expressions F_i of the MVL switches
- Expressions minimization

Phase 1:Each path leading to a leaf "i", gives a unique combination of input variables. Or-ing the corresponding expressions we get the expression for the switch F_i, as interconnections of unary subswitches.

For example, for MDD in figure 4, the values of variables on each path are:

- Paths to leaf 0:
  x1 x2 = 01, 11, and 22
- Paths to leaf 1:
  x1 x2 = 02 and 20
- Paths to leaf 2:
  x1 x2 = 00, 10, 12 and 21

As the result, the expressions for switches are:

$F_0(x1, x2) = x_1^0(0) \bullet x_2^1(0) + x_1^1(0) \bullet x_2^1(0) + x_1^2(0) \bullet x_2^2(0)$
$F_1(x1, x2) = x_1^2(1) \bullet x_2^0(1) + x_1^0(1) \bullet x_2^2(1)$
$F_2(X1, X2) = x_1^0(2) \bullet x_2^0(2) + x_1^1(2) \bullet x_2^0(2) +$
$x_1^2(2) \bullet x_2^1(2) + x_1^1(2) \bullet x_2^2(2)$

Phase 2: Circuit minimization.

In this phase, we try to find the equivalent expressions for switches Fi so that the number of transistors is lowered.

We build cost-table that describes the cost of each unary subswitch, i.e. the number of transistors contained in each subswitch. The Table 2 shows the ternary implementation with CMOS transistors (Hassan 1996)

Then, we represent each switch Fi as an expression tree. In that tree the leaves represent unary subswitches and internal nodes denote subswitches interconnections. In figure 5 is the tree for F_0. The symbol "•" denotes a series, "+" a parallel and "#" cascade interconnection.

We define the cost of that tree recursively, as below:

- The cost of a leaf (corresponding to an unary subswitch) is the number transistors on that subswitch(from Table 2)
- The cost of an internal node is the sum of the children subtrees.

SubSwitch	Cost(nr of transistors)
$x^0(0)$	3
$x^1(0)$	5
$x^2(0)$	6
$x^0(1)$	1
$x^1(1)$	6
$x^2(1)$	1
$x^0(2)$	6
$x^1(2)$	5
$x^2(2)$	2
$x^2(1,2)=x^{0,1}(2,1)$	2
$x^0(1,0)=x^{1,2}(0,1)$	2
$x^2(0,2)=x^{0,1}(2,0)$	3
$x^0(2,0)=x^{1,2}(0,2)$	3

Table 2 Cost table

The recursive function COST below returns the cost for a node in the tree:

```
int COST (TreeNode node)
{
  if Leaf (node) then
      return (cost from tabele 2 )
  else
      return (sum of costs of subtree of node)
}
```

For example, in figure 5 is the tree corresponding to F_0

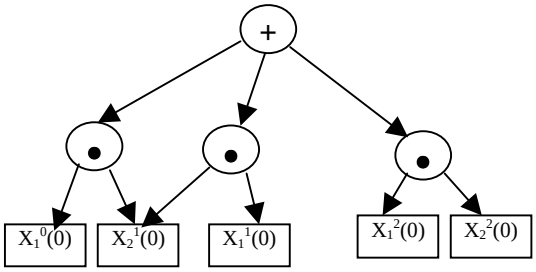

Fig 5. Expression tree for F_0. Cost(F_0)=30 and the cost of the tree is 30.

From algebraic properties of the interconnection operators (Hassan 1996), we found rules that, applied to the tree nodes, decrease the cost of the tree.

- Distributive law of series-parallel interconnections is depicted in figure 6.
- Rule R1, in figure 7, represents the series interconnection property
- Rule R2, in figure 8, represents the parallel interconnection property
- Rule R3, in figure 9, results from cascade interconnection definition.

For example, using distributive law, the tree for F_0 on figure 5, become as in figure 9. The cost of resulting tree is now 25. Applying other rules as follow, we have:

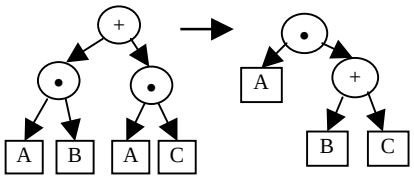

Fig. 6. Series-Parallel distributive law

$x^0(0) + x^1(0) = x^{0,1}(0)$ (rule R2).
$x^{0,1}(0) = x^{0,1}(2,0) \# x^{1,2}(0)$ (rule R3).
Rule R3 was applied because, from Table 2:
$Cost(x^{0,1}(0)) = 8$ and $Cost(x^{0,1}(2,0) \# x^{1,2}(0)) = 4$

The resulting tree is shown in figure 10. This is the best value for F_0: $Cost(F_0)=21$

Finally we obtain the minimal expressions for the three switches:
$F_0(x_1,x_2) = [x_1^{0,1}(2,0) \# x_1^{1,2}(0)] \bullet x_2^1(0) + x_1^2(0) \bullet x_2^2(0)$
$F_1(x_1,x_2) = x_1^2(1) \bullet x_2^0(1) + x_1^0(1) \bullet x_2^2(1)$
$F_2(x_1,x_2) = x_1^{0,1}(2) \bullet x_2^0(2) + x_1^2(2) \bullet x_2^1(2) + x_1^1(2) \bullet x_2^2(2)$

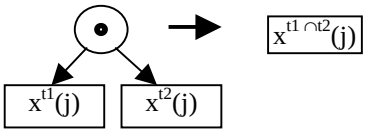

Fig. 7 Rule R1: series interconnections

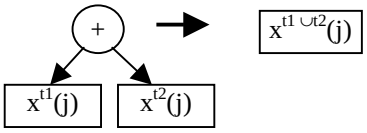

Fig. 8 Rule R2: parallel interconnections

The Optimisation Algorithm is:

```
TreeNode Optimisation (TreeNode   node){
  //  terminal case: an leaf
  if (Leaf(node))    return OptimalImplementation(node);

  if (R1)  return (Optimisation (right part R1) )
  if (R2)  return (Optimisation(riht part R2) )
  if (distributive law)
      applyDistributivity(node) // fig. 6

  //recursive call on each subtree
  for ( each child_i of node){
    node.child(i)= Optimisation (child_i);
  }
}
```

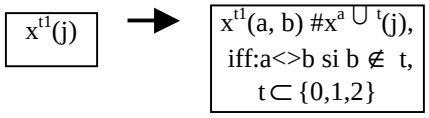

Fig. 9 Rule R3

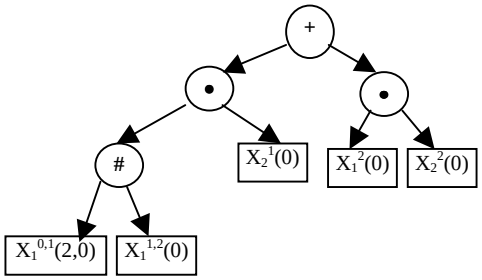

Fig.10 Final tree for F0. Cost (F0)=21

CONCLUSIONS

In (Hassan 1996) is shown that the MVL switches could offer a better solution for the MVL systems. In the above paper the authors have used only manual minimisation technique for logic minimisation. The algorithm described in this paper complements the work done in the cited paper, giving a systematic algorithm that simplifies the minimisation techniques.

REFERENCES

Drechsler, R. , 1998- Verification of Multi-Valued Logic Networks -Multiple-Valued Logic - An International Journal, Volume 3, pp. 77-88

Hassan M. et al., 1996–A Framework for Design a Multivalued Logic Functions and Its Application Using CMOS ternary Switches – IEEE Transactions on Circuits and Systems, Vol. 43 No.4, Apr.

Kam T. , 1995-State Minimization of Finite State Machines using Implicit Techniques, - PhD dissertation, ic.EECS.Berkeley.EDU