Methods of Removing Single Variable Static Hazards in Boolean Functions

Richard S. Sandige, *Senior Member, IEEE,* and Bogdan M. Wilamowski, *Senior Member, IEEE*

***Abstract*— Two methods are presented in this paper for removing single variable static hazards in Boolean Functions. The first method is a hand calculation method that uses a single variable static hazard cover algorithm. The hand calculation method can be used in the classroom and does not require computer assistance. The second method involves using a single variable static hazard cover software synthesis tool. The second method is more convenient to use outside the classroom on a PC that is available for student use or perhaps on personal computers. To illustrate the process of obtaining single variable static hazard covers, the standard Karnaugh map approach is also used. A commercially available design synthesis tool and schematic capture/simulation timing tool are also used in this paper.**

I. Introduction

STUDENTS in electrical and computer engineering curriculums need to know how to obtain single variable static (or logic) hazard covers [1] in order to design predictable asynchronous logic circuits. The hand calculation method provides good insight and thus a different method than the standard Karnaugh map method [2] for determining logic hazard covers. A logic hazard can produce a momentary malfunction called a glitch [3] in a circuit implementation of a Boolean function as a result of a single input variable change. A logic glitch produced by a logic hazard occurs when a single signal change progresses along at least two different delay paths that have unequal time delays to the same output, and each signal path can force the output to a different value [4]. The two methods presented in this paper for removing logic hazards are not limited by the number of input variables. In comparison, the standard Karnaugh method is only useful for perhaps up to six variables because of the difficulty in drawing and reading larger map sizes.

Four and five variable Boolean functions are presented in Section II to illustrate the hand calculation method as compared to the standard Karnaugh map method. The software synthesis tool PLDesigner-XL [5] is used to minimize Boolean functions while the schematic capture/ simulation timing software tool B^{2}Logic [6] is used to show the presence of glitches caused by logic hazardous functions and to verify the removal of glitches for logic hazard free functions. Both the hand calculation method and the Karnaugh map method are somewhat time consuming; however, the software synthesis tool presented in Section III is an automated logic hazard cover tool that is both easy to use as well as a time saver.

Manuscript received September 29, 1993; revised April 4, 1994.

The authors are with the Department of Electrical Engineering, University of Wyoming, Laramie, WY 82071 USA.

IEEE Log Number 9413597.

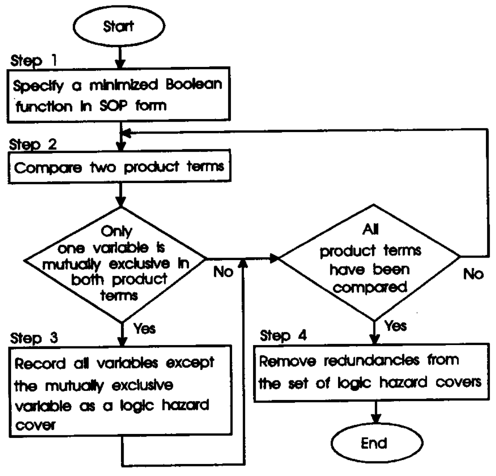

Start
Step 1
Specify a minimized Boolean function in SOP form
Step 2
Compare two product terms
Only one variable is mutually exclusive in both product terms
No
Yes
Step 3
Record all variables except the mutually exclusive variable as a logic hazard cover
All product terms have been compared
No
Yes
Step 4
Remove redundancies from the set of logic hazard covers
End

Fig. 1. A logic hazard cover algorithm.

II. Solving for Logic Hazards Using the Hand Calculation Method

The hand calculation method involves applying the logic hazard cover algorithm illustrated in Fig. 1 to a Boolean function. Consider the following Boolean function:

$$\mathrm{F1}(W,X,Y,Z) = \sum m(0,2,3,4,8,9,10,14) \qquad (1)$$

Step 1 of the algorithm requires that a Boolean function be in a minimized sum of products (SOP) form. An SOP form of (1) can easily be obtained using a software synthesis tool that results in a minimized SOP form of the function. A design file for the software synthesis tool PLDesigner-XL is first created as shown in Fig. 2(a) in terms of the minterms of the ones of the function. Configuring a) global options for Quine-McCluskey reduction [7] and b) documentation options for Print DeMorgan Equations, then "building" or compiling the design file, provides the minimum SOP forms of (1) as shown in Fig. 2(b). A Boolean function can be written in the design file in any algebraically correct form, and this program will obtain the minimum SOP forms for the ones and for the zeros of the function. Inverting both sides of the equation for the SOP form for the zeros of a function results in an equation for the POS form of the function. This is useful to know if it is desired to use a NOR-NOR implementation for the function. Logic hazard covers can be calculated for the ones or the zeros of a function via the logic hazard cover algorithm. Using the minimized SOP form for the zeros

```
INPUT W,X,Y,Z;
OUTPUT F1;

F1 = /W*/X*/Y*/Z  + /W*/X*Y*/Z + /W*/X*Y*Z  + /W*X*/Y*/Z
   + W*/X*/Y*/Z +  W*/X*/Y*Z +  W*/X*Y*/Z +  W*X*Y*/Z;
```

(a)

```
EQUATIONS FOR SYSTEM
INPUT SIGNALS (4) :
        W
        X
        Y
        Z
OUTPUT SIGNALS (1) :
        F1
REDUCED EQUATIONS:
F1.EQN = /W*/Y*/Z
           + W*Y*/Z
           + W*/X*/Y
           + Y*/W*/X ;   "(4 terms, 4 symbols)
F1.EQN(~) = Z*/W*/Y
           + W*Y*Z
           + W*X*/Y
           + X*Y*/W ;   "(4 terms, 4 symbols)
```

(b)

Fig. 2. (a) Software synthesis tool PLDesigner-XL design file for (1), (b) minimum SOP equations obtained after compiling the design file.

Steps 2 and 3:

$$\overline{F1} = Z\cdot\overline{W}\cdot\overline{Y} + W\cdot Y\cdot Z + W\cdot X\cdot\overline{Y} + X\cdot Y\cdot\overline{W}$$

none

$\mathrm{LHC} = X\cdot\overline{Y}\cdot Z$

$\mathrm{LHC} = \overline{W}\cdot X\cdot Z$

$\mathrm{LHC} = W\cdot X\cdot Z$

$\mathrm{LHC} = X\cdot Y\cdot Z$

none

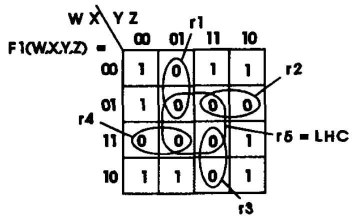
Step 4:

$$\begin{aligned}\mathrm{LHC} &= X\cdot\overline{Y}\cdot Z + \overline{W}\cdot X\cdot Z + W\cdot X\cdot Z + X\cdot Y\cdot Z\\ &= X\cdot(\overline{Y} + Y)\cdot Z + (\overline{W} + W)\cdot X\cdot Z\\ &= X\cdot Z + X\cdot Z\\ &= X\cdot Z\end{aligned}$$

Fig. 3. Calculation of the logic hazard covers of the zeros of (1) using the hand calculation method.

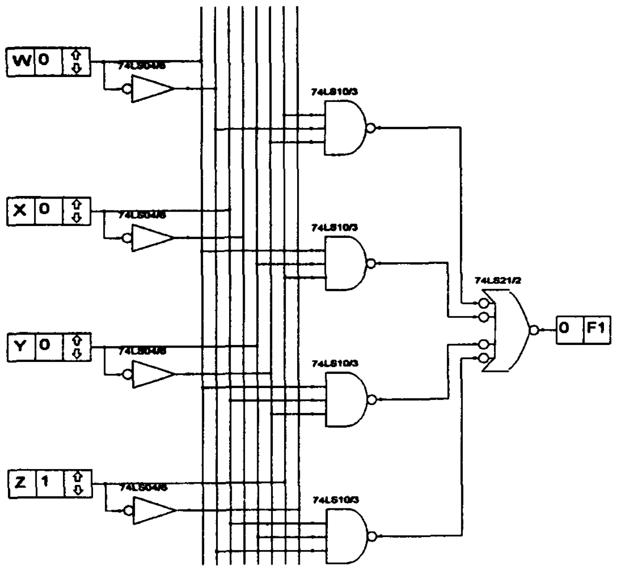
of (1) (the inverted POS form), represented in Fig. 2(b) as F1.EQN(~), the logic hazard covers are calculated using the hand calculation method for steps 2 and 3 as shown in Fig. 3. "Only one mutually exclusive variable in both product terms" in the decision box in Fig. 1 means a single variable and its complement. If more than one variable and its complement exist when two product terms are compared, then the No path is taken; otherwise the Yes path is taken. A logic hazard cover in Step 3 consists of all the terms in both product terms that were compared ANDed together omitting the single variable and its complement, i.e., the mutually exclusive variable. Since more than one logic hazard cover exists, step 4 requires that redundancies be removed from the set of logic hazard covers. Redundancies are removed from the set of logic hazard covers by minimizing the logic hazard cover set then removing any remaining logic hazard cover terms that are also included in the set of product terms that make up the minimum SOP form of the function. After applying step 4 the only logic hazard cover remaining is X·Z.

To verify this result, the Karnaugh map method for obtaining logic hazard covers is illustrated in Fig. 4. When an SOP

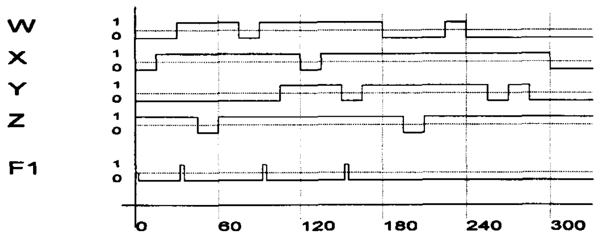


$$\begin{aligned}\overline{F1} &= r1 + r2 + r3 + r4 + r5\\ &= \overline{W}\cdot\overline{Y}\cdot Z + \overline{W}\cdot X\cdot Y + W\cdot Y\cdot Z + W\cdot X\cdot\overline{Y} + X\cdot Z\end{aligned}$$

Fig. 4. Calculation of the logic hazard covers of the zeros of (1) using the Karnaugh map method.



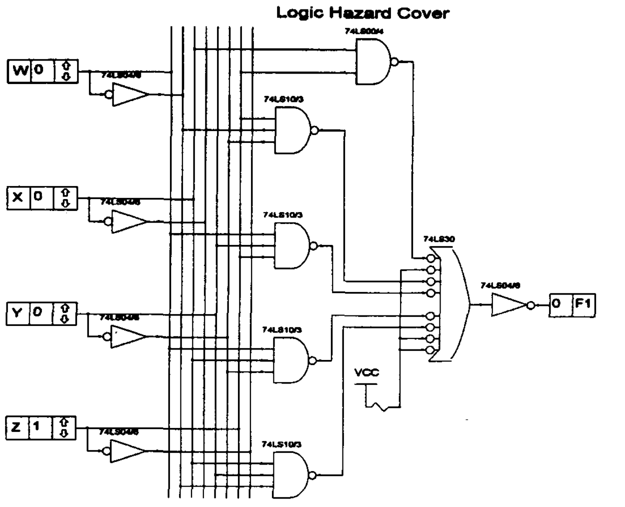
(a)



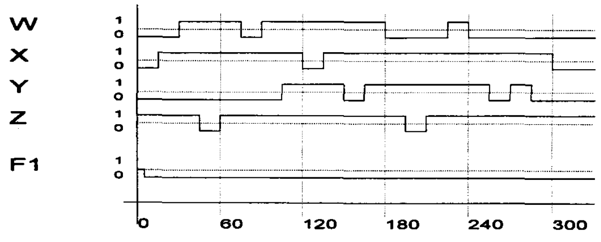
(b)

Fig. 5. (a) Circuit diagram for a minimum SOP form of the zeros of (1) using the software package B^2logic, (b) B^2logic simulation timing diagram showing logic 1 glitches.

function is implemented with missing logic hazard covers, the output signal can contain logic glitches. Fig. 5(a) show the circuit diagram for a minimized SOP function for (1) while Fig. 5(b) illustrates a timing simulation of the circuit containing logic 1 glitches. Notice that a logic 1 glitch is present at approximately 30 ns and again at 90 ns when input *W* changes from 0 to 1 with all other inputs stable. Another logic 1 glitch is present at approximately 150 ns when input *Y* changes from 1 to 0 with all other inputs stable. These logic



(a)



(b)

Fig. 6. (a) Circuit diagram for a minimum SOP form of the zeros of (1) including the logic hazard cover term using the software package B^{2}logic, (b) B^{2}logic simulation timing diagram showing the elimination of logic 1 glitches.

glitches in the timing simulation of the circuit are due to the absence of the logic hazard cover term in the circuit. When an output should stay at the value 0 and it momentarily changes to the value 1 and then back to the value 0 due to a single variable changing its value, the hazard is often called a static 0 logic hazard, and the glitch that is produced is called a logic 1 glitch. The logic 1 glitch can be removed by simply adding the required logic hazard cover term.

As discussed earlier, the required logic hazard cover terms can be obtained by manually applying the logic hazard cover algorithm or by utilizing the Karnaugh map method. When the logic hazard cover term is added to the function as shown in the schematic in Fig. 6(a), the logic 1 glitches are eliminated as seen by the timing simulation displayed in Fig. 6(b). The circuit is now logic hazard free (LHF) since the circuit is logic glitch free (LGF).

As another example consider the following five-variable Boolean function.

$$F2(V, W, X, Y, Z) = \sum m(1,3,4,6,7,10,11,12,14,19,20,26,27,28). \quad (2)$$

The design file for (2) is shown in Fig. 7(a). Minimum SOP forms of (2) that are obtained after the design is compiled are shown in Fig. 7(b) thus completing step 1. Equation F2.EQN

```
INPUT V,W,X,Y,Z;
OUTPUT F2;

F2 = /V*/W*/X*/Y*Z + /V*/W*/X*Y*Z + /V*/W*X*/Y*/Z + /V*/W*X*Y*/Z
     + /V*/W*X*Y*Z + /V*W*/X*Y*/Z + /V*W*/X*Y*Z   + /V*W*X*/Y*/Z
     + /V*W*X*Y*/Z + V*/W*/X*Y*Z  + V*/W*X*/Y*/Z  + V*W*/X*Y*/Z
     + V*W*/X*Y*Z  + V*W*X*/Y*/Z;
```

(a)

```
EQUATIONS FOR SYSTEM
INPUT SIGNALS (5) :
        V
        W
        X
        Y
        Z
OUTPUT SIGNALS (1) :
        F2
REDUCED EQUATIONS:
F2.EQN = Z*/V*/W*/X
                  + X*Y*/V*/W
                  + X*/Y*/Z
                  + X*/V*/Z
                  + Y*Z*/X
                  + W*Y*/X ;   "(6 terms, 5 symbols)
F2.EQN(~) = /W*/X*/Z
                  + X*Z*/Y
                  + W*X*Z
                  + W*/X*/Y
                  + V*/X*/Y
                  + V*X*Y ;   "(6 terms, 5 symbols)
```

(b)

Fig. 7. (a) Software synthesis tool PLDesigner-XL design file for (2), (b) minimum SOP equations obtained after compiling the design file.

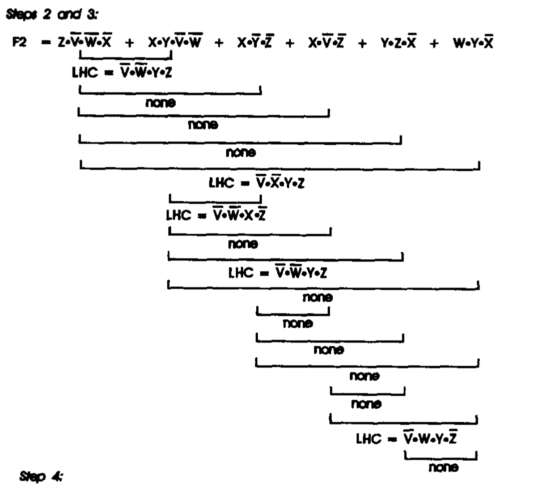
in Fig. 7(b) represents a minimized SOP form for the ones of (2). The logic hazard covers are calculated using the hand calculation method for steps 2 and 3 as shown in Fig. 8. Since multiple logic hazard covers exist, any redundancies in the set of logic hazard covers should be removed as shown in Fig. 8 under step 4. Only two logic hazard cover terms remain after applying step 4. A logic hazard free function for (2) consists of the minimum SOP form for the ones of the function with the two logic hazard covers added.

As a comparison, the same result is obtained in Fig. 9 using the Karnaugh map method. As the number of input variables gets larger, both the hand calculation method and the Karnaugh map method become very time consuming. The application of the software synthesis tool in Section III speeds up the process of calculating logic hazard covers so that logic hazard free equations can be easily obtained for asynchronous sequential designs.

III. Solving for Logic Hazards Using the Synthesis Tool

A more efficient method of obtaining required logic hazard cover terms is to write a computer program to automate the logic hazard cover algorithm. A software synthesis tool called HAZARD is a computer program written to perform the logic hazard cover algorithm presented in Fig. 1 in an automated fashion. The program was written in PASCAL and runs on an IBM PC or compatible. First, a text file must be prepared that contains the product terms for a minimum SOP function such as the text file shown in Fig. 10(a) for (1) for the zeros of F1. Any word processor or text editor that creates an ASCII file can be used. The extension for the file name must be .LIN (which stands for Logic INput). The program is run by typing HAZARD, pressing Enter, typing the name assigned to the

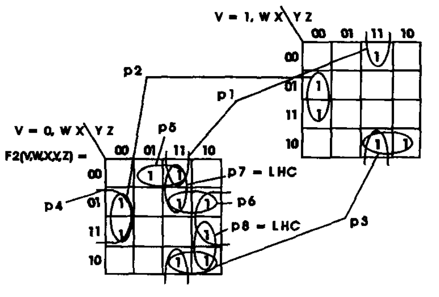
Steps 2 and 3:



Step 4:

$$
\begin{aligned}
\text{LHC} &= \bar{V}\cdot\bar{W}\cdot Y\cdot Z + \bar{V}\cdot\bar{X}\cdot Y\cdot Z + \bar{V}\cdot\bar{W}\cdot X\cdot\bar{Z} + \bar{V}\cdot\bar{W}\cdot Y\cdot Z + \bar{V}\cdot W\cdot Y\cdot\bar{Z}\\
&= \bar{V}\cdot\bar{W}\cdot Y\cdot Z + \underset{\substack{\uparrow\\ \text{part of}\\ Y\cdot Z\cdot\bar{X}}}{\bar{V}\cdot\bar{X}\cdot Y\cdot Z} + \underset{\substack{\uparrow\\ \text{part of}\\ X\cdot\bar{V}\cdot\bar{Z}}}{\bar{V}\cdot\bar{W}\cdot X\cdot\bar{Z}} + \bar{V}\cdot W\cdot Y\cdot\bar{Z}\\
&= \bar{V}\cdot\bar{W}\cdot Y\cdot Z + \bar{V}\cdot W\cdot Y\cdot\bar{Z}
\end{aligned}
$$

Fig. 8. Calculation of the logic hazard covers of the ones of (2) using the hand calculation method.



$$
\begin{aligned}
F2 &= p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8\\
&= \bar{X}\cdot Y\cdot Z + X\cdot\bar{Y}\cdot\bar{Z} + W\cdot\bar{X}\cdot Y + \bar{V}\cdot X\cdot\bar{Z} + \bar{V}\cdot\bar{W}\cdot\bar{X}\cdot Z\\
&\quad + \bar{V}\cdot\bar{W}\cdot X\cdot Y + \bar{V}\cdot\bar{W}\cdot Y\cdot Z + \bar{V}\cdot W\cdot Y\cdot\bar{Z}
\end{aligned}
$$

Fig. 9. Calculation of the logic hazard covers of the ones of (2) using the Karnaugh map method.

text file such as F1.LIN, and then pressing Enter. The input data (the product terms for a minimum SOP function) and the logic hazard covers (calculated via the logic hazard cover algorithm) are shown on the screen as illustrated in Fig. 10(b) and also stored in F1.OUT. Notice that the calculated logic hazard covers for function F1 are the same as those obtained in Section II. To obtain help in using the program, the user can type HAZARD, type Enter, and then press function key F1.

Fig. 11(a) shows the text file F2.LIN that was created for (2) for the ones of the function F2. After running the software synthesis tool HAZARD, the input data and the calculated logic hazard covers for function F2 are displayed as shown in Fig. 11(b). As expected, the logic hazard covers are the same as those obtained in Section II using the hand calculation method and the Karnaugh map method. As the number of input variables increases, the software synthesis tool can save a lot

```
Z/W/Y
WYZ
WX/Y
XY/W
```

(a)

```
•  input data:
/W/YZ
WYZ
WX/Y
/WXY
*  logic hazard covers:
XZ
```

(b)

Fig. 10. (a) Text file for a minimum SOP form of the zeros of F1, (b) calculation of the logic hazard covers of the zeros of F1 using the software synthesis tool HAZARD.

```
Z/V/W/X
XY/V/W
X/Y/Z
X/V/Z
YZ/X
WY/X
```

(a)

```
•  input data:
/V/W/XZ
/V/WXY
X/Y/Z
/VX/Z
/XYZ
W/XY
•  logic hazard covers:
/V/WYZ
/VWY/Z
```

(b)

Fig. 11. (a) Text file for a minimum SOP form of the ones of F2, (b) calculation of the logic hazard covers of the ones of F2 using the software synthesis tool HAZARD.

of time in calculating logic hazard covers as demonstrated by these examples. The synthesis tool can also be used to verify the elimination of all logic hazards by rerunning HAZARD with all logic hazard covers included with the minimum SOP form of the function. If the software synthesis tool indicates that no logic hazard covers are required, then all logic hazards have been eliminated. This is illustrated in Fig. 12(a) and 12(b) for function F1 and F2 respectively.

IV. Conclusion

This paper has presented two methods for calculating logic hazard covers. The first method was a hand calculation method. This method can be used by students in a classroom situation as a substitute for the standard Karnaugh map method. Both the hand calculation method and the software synthesis tool use the logic hazard cover algorithm. The hand calculation method is time consuming but can generally be applied very easily even for Boolean functions that contain a larger number of input variables. The software synthesis tool is more convenient to use, and saves time in calculating logic hazard covers. The design synthesis tool PLDesigner-XL was used to obtain minimized SOP forms of Boolean functions. The schematic capture/simulation timing tool B^2Logic was used to show the presence of logic glitches caused by logic hazardous functions and to verify the removal of logic glitches

```
•   input data:
/W/YZ
WYZ
WX/Y
/WXY
XZ
*  no logic hazard covers
```

(a)

```
*   input data:
/V/W/XZ
/V/WXY
X/Y/Z
/VX/Z
/XYZ
W/XY
/V/WYZ
/VWY/Z
*  no logic hazard covers
```

(b)

Fig. 12. (a) Result of rerunning the software synthesis tool HAZARD to verify the elimination of all logic hazards of F1 when all logic hazard covers are included with a minimum SOP form of the function, (b) result of rerunning the software synthesis tool HAZARD to verify the elimination of all logic hazards of F2 when all logic hazard covers are included with a minimum SOP form of the function.

for logic hazard free functions. The software synthesis tool HAZARD was also used to verify the elimination of logic hazards. The software synthesis tool HAZARD is a free-ware program available from the authors.

REFERENCES

[1] R. F. Tinder, *Digital Engineering Design—A Modern Approach.* Englewood Cliffs, NJ: Prentice-Hall, 1991.

[2] R. S. Sandige, *Modern Digital Design.* New York: McGraw-Hill, 1990.

[3] C. Innes, "Avoiding programmable logic hazards," *National Anthem, National Semiconductor*, p. 5, Jan./Feb. 1988,

[4] S. H. Unger, *The Essence of Logic Circuits.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[5] *PLDesigner-XL^{TM}, The Next Generation In Programmable Logic Design Synthesis, Version 3.0, User's Guide,* Minc Incorporated, 1992.

[6] J. A. Engelbert, *User Manual for B^2 Logic v2.2.* Ann Arbor, MI: Beige Bag (B^2) Software, 1990.

[7] E. J. McCluskey, *Logic Design Principles with Emphasis of Testable Semicustom Circuits.* Englewood Cliffs, NJ: Prentice-Hall, 1986.

Richard S. Sandige (S'63–M'64–SM'90) received the B.S.E.E. and M.S.E.E. degrees from West Virginia University, Morgantown, in 1963 and 1969, respectively, and the Ph.D. degree in electrical engineering from Texas A&M University, College Station, in 1978.

Currently, he is a Full Professor in the Department of Electrical Engineering, University of Wyoming. He has written several books and papers. His latest book is *Modern Digital Design*, McGraw-Hill, 1990. Prior to coming to the University of Wyoming he was a member of the Technical Staff in the R&D Laboratory at Hewlett Packard, Fort Collins, CO. His research interests lie in the areas of PLD's and FPGA's and their applications, digital design, HDL's, microprocessor systems, and fuzzy logic. He currently supports the research and development activities of Minc Incorporated.

Bogdan M. Wilamowski (SM'83) received his M.S. in computer engineering in 1966, his Ph.D. in neural computing in 1970, and his D.Sc. in integrated circuit design in 1977, all from Technical University of Gdansk, Poland.

From 1968 to 1970 he was with the Nishizawa Laboratory at Tohoku University, Japan. In 1975–1976, he spent one year at the Semiconductor Research Institute, Sendai, Japan as JSPS fellow. He became an Associate Professor at the Technical University of Gdansk in 1978 and Professor in 1987 and from 1979 to 1981, he was the Director of the Institute of Electronic Technology. During 1981 and 1982, he was a faculty member at Auburn University. From 1982 to 1984 he was a Visiting Professor at the University of Arizona, Tucson. From 1988 to 1989 he was the Head of the Solid State Electronic Chair at the Technical University of Gdansk. He came to the University of Wyoming in 1989, and he is currently a Full Professor in the Electrical Engineering Department. Dr. Wilamowski is the author of 2 books, more than 150 refereed publications, and holds about 30 patents in 7 countries. His research interests lie in the areas of semiconductor devices, electronic, integrated circuits, and computer simulation and modeling.