An Intuitive Visual Method for Introducing Boolean Equation Reduction Using PLC Ladder Logic Notation

Allen E. Leybourne
University of Southern Mississippi

Introduction

Boolean algebra has been known for centuries, but alas, students exposed to these materials for the first time must develop their understanding from the beginning. One obstacle is they have probably already learned traditional algebra in their earlier academic passage. Clearly, Boolean algebra is different. But why? More time can be spent on this question than might at first be imagined. The question is usually addressed through a number of Boolean equation derivations, drills with truth tables, and the time honored Karnaugh (K) map. Paraphrasing the words of one textbook author [1]; once a Boolean equation is in the sum of products (SOP) form, it can be placed onto the K map which produces an almost "cookbook" approach for reduction of the equation to its simplest form. Although this avoids the cleverness (requirement for intuitive discovery) often required for Boolean algebra based approaches, one must understand the rules for construction of the K map and its use. The Quine-McMclusky method can be viewed as a tabular approach to Boolean equation simplification that has significant visual impact on its user [2].

Interestingly, commercial organizations which have successfully marketed programmable logic controllers (PLCs) adopted the ladder logic language as the programming interface for these Boolean machines because of the simplicity of its interpretation. It will become clear during this paper that the commercial choice in favor of the ladder logic language was no accident. A visually intuitive technique for the discovery of reduced Boolean equation forms using the PLC ladder logic notation is described.

Techniques for Introducing Boolean Algebra Reduction

1. Of course this topic can be introduced devoid of any applications using the time honored techniques of mathematical proofs.

2. Karnaugh maps, which limit visual analysis to about five Boolean variables, use a set of rules for generation of reduced equations. In some respects, the development of these rules is as arcane as the Boolean algebra itself. Also, any error made in generating the K map will propagate errors into the solution.

3. Truth tables, one basis for the Quine-McClusky method, can be used to develop redundant true outputs. Through a somewhat convoluted logic (at least for the novice) it is possible to reduce the SOP equation to a simpler form.

4. There are also other techniques, one of which is described in this paper.

A Comparison of the methods - Example Case for two Boolean variables A and B.

Let us start with a simple 2 variable SOP case with the Boolean equation: $Y=AB+\bar{A}B+A\bar{B}+\bar{A}\bar{B}$.

The Truth table for: $Y=AB+\bar{A}B+A\bar{B}+\bar{A}\bar{B}$				
A	B	$\bar{A}$	$\bar{B}$	Y=
0	0	1	1	$\bar{A}\bar{B}$ + or 1+
0	1	1	0	$\bar{A}B$ + or 1+
1	0	0	1	$A\bar{B}$ + or 1+
1	1	0	0	AB or 1

From this truth table, the output Y = 4 = 1 or TRUE for all state combinations of A and B.

The K map for this case, $Y=AB+\bar{A}B+A\bar{B}+\bar{A}\bar{B}$, is:

	$\bar{B}$	B
$\bar{A}$	1	1
A	1	1

The rules for equation element reduction after the equation is in SOP form is to place a 1 in appropriate cells of the K map. Now group the K map elements which have adjacent cells containing a 1 (including those with toroidal topology). Upon simplification, Y=1. In all, two to three pages are required to describe the rules for generating a K map and reducing the equation through its use. One annoyance is the procedure required before the Boolean equation can plotted on the K map. Using the example Y=A+B; in order to place it on the K map, extra terms must be generated. In this case: $Y=A+B=A(B+\bar{B})+(A+\bar{A})B= AB +A\bar{B} +AB+\bar{A}B$. The extra terms generated for plotting on the K map will eventually be reduced, but why complicate the equation and then reduce it? In response, efficient computer codes have been written which easily follow the complex K map rules. However, this intermediate expansion is a difficult concept to explain and justify to new students.

The ladder logic diagram has very simple rules. One assumes that power, the true (T) state, can flow only through the Boolean variables (contact) when in a true state and that reverse flows are not permitted. For those unfamiliar with ladder logic programming, the backslash, [\], denotes the complement of the Boolean variable. Ladder rung structures similar to the following example are created for each of the SOP terms. The [ ] and ( ) denote input and outputs, respectively.

```
T
|-[ ]---[\]---[ ]---[\]---( )--
   A     A     B     B     Y
```

Applying these procedures to the Boolean Equation, one obtains the **Generation 1** ladder diagram representation.

```
T
|---[ ]-------[ ]-------( )--
|    A         B    |    Y
|                   |
|---[ ]-------[\]---|
|    A         B    |
|                   |
|---[\]-------[ ]---|
|    A         B    |
|                   |
----[\]-------[\]----
     A         B
```

Reduction to **Generation 2** is relatively obvious.

```
T
|---[ ]-------[ ]-------( )--
|    A    |    B    |    Y
|         |         |
|         ----[\]---|
|              B    |
|                   |
|---[\]-------[ ]---|
     A    |    B    |
          |         |
          ----[\]----
               B
```

The first impression here is **"A-ha!"**, I see what is happening. Reduction proceeds through visual analysis of the diagram (analogous to a circuit), to a simpler but non-the-less equivalent form. Of course, each subsequent generation must not change functionality. No rules are needed for diagram reduction as the results are quite intuitive when one views the logic as flowing through the Boolean variables for all possible state combinations. Proceeding on to **Generation 3.**

```
T
|--------------[ ]------( )--
           |    B    |   Y
           |         |
           ----[\]----
                B
```

Finally complete the analysis in **Generation 4**. Here we will note that Y is always true, i.e., Y=1 as was discovered using the other methods.

```
T
|-----------------------( )--
                          Y
```

This ladder logic approach has utilized the visual analytical power of our specie. The way in which the circuit works is obvious, even to one who has little or no background in the subject. Boolean algebraic concepts are ingrained almost effortlessly. The author admits that the initiate user of this method is still a long way from handling K maps and Boolean algebra, but is not that just the point.

For most students, it is important that their first impression of Boolean algebra is that it is useful and not just a weird new abstract algebra. This instructional approach is based solidly on a practical application. PLCs (Boolean machines) have ubiquitous uses today in industry. The PLC interface language is also very similar in appearance to the hardwired ladder logic diagrams used in industrial control panels, understood for decades by designers, plant engineers and plant electricians.

A Three Boolean Variable Example

The somewhat trivial example just presented illustrated the method but was of insufficient complexity to convey much besides the technique itself. A three variable SOP equation will be used to amplify understanding of the method. Reduction of the equation will be shown using two examples. The first produces a reduced equation, though not necessarily minimal, the second is a minimal equation.

Example 1 - Let $Y=\bar{A}BC+AB\bar{C}+ABC$, thus the diagram.

Generation 1

```
T                                  T or F
|-[\]-----[ ]------[ ]-----------( )--
|  A       B        C        |     Y
|                            |
|-[ ]-----[ ]------[\]-------|
|  A       B        C        |
|                            |
|-[ ]-----[ ]------[ ]--------
   A       B        C
```

Generation 2

```
T                                  T or F
|-[\]-----[ ]------[ ]-----------( )--
|  A       B        C        |     Y
|                            |
|-[ ]-----[ ]------[\]-------|
|  A       B    |   C        |
                |            |
                ---[ ]--------
                    C
```

Generation 3

```
T                                        T or F
|-[\]-----[ ]-----[ ]------------( )--
|  A       B       C          |     Y
|                             |
|-[ ]-----[ ]-----------------
   A       B
```

Generation 4

```
T                                        T or F
|-[ ]-----[\]------[ ]-----------( )--
   B   |   A        C         |     Y
       |                      |
       ---[ ]-----------------
           A
```

No further reduction is visually obvious. Thus the final reduced Boolean equation is: $Y=B(A+\bar{A}C)$.

Example 2 - The expansion step used in this example might not occur to the novice user, but once discovered, even though no proof is shown, it is visually clear that it makes sense.

Generation 1

```
T                                        T or F
|-[\]-----[ ]------[ ]-----------( )--
|  A       B        C         |     Y
|                             |
|-[ ]-----[ ]------[\]--------|
|  A       B        C         |
|                             |
|-[ ]-----[ ]------[ ]--------
   A       B        C
```

Generation 2

```
T                                        T or F
|-[ ]-----[\]------[ ]-----------( )--
|  B       A        C         |     Y
|                             |
|-[ ]-----[ ]------[\]--------|
|  B       A        C         |
|                             |
|-[ ]-----[ ]------[ ]--------
   B       A        C
```

Generation 3

```
T                                        T or F
|-[ ]-----[\]------[ ]-----------( )--
   B   |   A        C         |     Y
       |                      |
       |--[ ]------[\]--------|
       |   A        C         |
       |                      |
       ---[ ]------[ ]--------
           A        C
```

At this step it is clear that the interior part, ---[A]---[C]---, can be duplicated as shown without changing the output, Y. When using Boolean algebra, this is equivalent to using a term of the equation twice as different identities are sought.

Generation 4

```
T                                        T or F
|-[ ]-----[\]------[ ]-----------( )--
   B   |   A        C         |     Y
       |                      |
       |--[ ]------[ ]--------|
       |   A        C         |
       |                      |
       |--[ ]------[\]--------|
       |   A        C         |
       |                      |
       ---[ ]------[ ]--------
           A        C
```

Generation 5

```
T                                        T or F
|-[ ]-----[ ]------[\]-----------( )--
   B   |   C        A         |     Y
       |                      |
       |--[ ]------[ ]--------|
       |   C        A         |
       |                      |
       |--[ ]------[\]--------|
       |   A        C         |
       |                      |
       ---[ ]------[ ]--------
           A        C
```

Generation 6

```
T                                        T or F
 |-[ ]-----[ ]------[\]-----------( )--
    B   |   C   |    A         |     Y
        |       |              |
        |       ---[ ]---------|
        |           A          |
        |                      |
        ---[ ]------[\]--------|
            A   |    C         |
                |              |
                ---[ ]--------
                    C
```

Generation 7

```
T                                        T or F
|-[ ]-----[ ]--------------------( )--
   B   |   C                  |     Y
       |                      |
       ---[ ]-----------------
           A
```

No further reduction is visually obvious. Thus the final reduced Boolean equation is: Y=B(A+C). In this case further exploration would show that this is a minimal equation. It is believed that most students can follow examples such as the above without ever mentioning the rules of Boolean algebra or K maps. They can grasp the process because each step is visually obvious.

The tabular method of Quine-McClusky, described by Booth, can also be used to reduce Boolean equations. This method also utilizes a sort of visual analysis to simplify the Boolean equation and has the additional advantage of extension to a programming algorithm that can automate the process [2]. However, it is the author's opinion that the visualization process is much less intuitive than the one presented herein.

Summary

A method has been presented which allows students to reduce SOP Boolean algebra equations using visual analysis of the PLC ladder logic diagram. The reduction process proceeds by successive rearrangements of the diagram that are justified by common sense as conveyed by the visual approach. The author has used this method successfully in an automation class involving robotics, PLCs, and pneumatic logic circuits. The primary interests of these students are the applications of Boolean algebra rather than its theory. As the students become more experienced and motivated to seek value from Boolean algebra, those methods can be compared to the ladder logic.

One advantage is that students may start from the original equation without the proliferation of terms (whose justification is difficult to grasp) required by the K map method. When it is necessary to expand terms as in Example 2, the justification for the process is obvious. Though not discussed directly herein, it is believed that these techniques would assist students to comprehend the Quine-McMclusky method if the materials were introduced in parallel.

The main limitation of this method is the frequent redrawing of the diagrams for each generation which is tedious and subject to human error. For instructional purposes this is probably an acceptable limitation since core concepts can be presented clearly.

If space permitted, it could also be shown that the method can be extended to equations in product of sums (POSs.) form in a straight forward manner.

References

1. Floyd, Thomas L., Digital Fundamentals, 4th Ed, p. 164, (1990), Merrill Publishing Co.

2. Booth,Taylor L., Digital Networks and Computer Systems, 2nd Ed., p. 192, (1978), John Wiley and Sons.

ALLEN E. LEYBOURNE, III

Allen Leybourne received the BS and PhD degrees in Chemical Engineering from the University of Florida in 1956 and 1961 respectively. He obtained an MS degree in Chemical Engineering from Pennsylvania State University in 1958. He has over 15 years experience in polyester research, development, and manufacturing and over 10 years industrial experience in facility design and implementation. As a contractor to Interpine Lumber; he designed, built, and installed a fully automated wood-fired dry kiln that was totally controlled by an advanced level Programmable Logic Controller (PLC). Allen is now a professor in the School of Engineering Technology, having served nearly 15 years. He has taught courses in digital logic, process automation, PLCs and robotics.