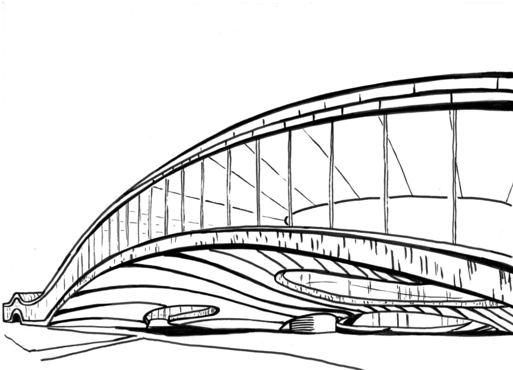
The fascinating properties of majority

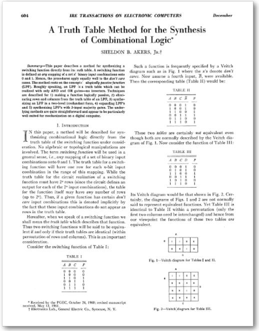
Mathias Soeken

Integrated Systems Laboratory, EPFL, Switzerland

mathias.soeken@epfl.ch   msoeken.github.io   msoeken/cirkit



A brief history of majority logic



A Truth Table Method for the Synthesis of Combinational Logic*

SHELDON B. AKERS, JR.†

Summary—This paper describes a method for synthesizing a switching function directly from its truth table. A switching function is defined as *any mapping* of a set of binary input combinations onto 0 and 1. Hence, the procedures apply equally well to the *don't care* cases. The method rests on the concept of a *logically passive function* (LPF). Roughly speaking, an LPF is a truth table which can be realized with only AND and OR gates—no inverters. Techniques are described for 1) making a function logically passive, 2) eliminating rows and columns from the truth table of an LPF, 3) synthesizing an LPF in a two-level irredundant form, 4) expanding LPF's and 5) synthesizing LPF's with 3-input majority gates. The underlying methods are quite straightforward and appear to be particularly well suited for mechanization on a digital computer.

I. Introduction

IN this paper, a method will be described for synthesizing combinational logic directly from the truth table of the switching function under consideration. No algebraic or topological manipulations are involved. The term *switching function* will be used in a general sense, *i.e.*, *any* mapping of a set of binary input combinations onto 0 and 1. The truth table for a switching function will have one row for each n-bit input combination in the range of this mapping. While the truth table for the circuit *realization* of a switching function must have 2^n rows (since the circuit defines an output for each of the 2^n input combinations), the table for the function itself may have *any* number of rows (up to 2^n). Thus, if a given function has certain *don't care* input combinations this is denoted implicitly by the fact that these input combinations do not appear as rows in the truth table.

Hereafter, when we speak of a switching function we shall mean the *truth table* which describes that function. Thus two switching functions will be said to be equivalent if and only if their truth tables are identical (within permutation of rows and columns). This is an important consideration.

Consider the switching function of Table I:

TABLE I

A	B	C	F
0	0	0	0
1	0	0	0
1	1	0	1
0	0	1	1
0	1	1	0
1	1	1	1

Such a function is frequently specified by a Veitch diagram such as in Fig. 1 where the x's denote *don't cares*. Now assume a fourth input, $\bar{B}$, were available. Then the corresponding table (Table II) would be:

TABLE II

A	B	C	$\bar{B}$	F
0	0	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	1
0	1	1	0	0
1	1	1	0	1

These two *tables* are certainly *not* equivalent even though both are normally described by the Veitch diagram of Fig. 1. Now consider the function of Table III:

TABLE III

A	B	C	D	F
0	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	1
1	0	1	0	0
1	1	1	0	1

Its Veitch diagram would be that shown in Fig. 2. Certainly, the diagrams of Figs. 1 and 2 are not normally said to represent equivalent functions. Yet Table III is identical to Table II within a permutation (only the first two columns need be interchanged) and hence from our viewpoint the functions of these two tables *are* equivalent.

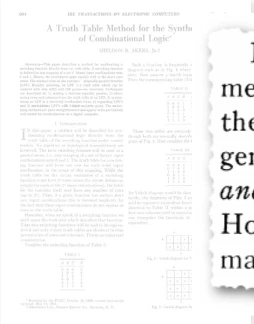
Fig. 1—Veitch diagram for Tables I and II.

Fig. 2—Veitch diagram for Table III.

* Received by the PGEC, October 24, 1960; revised manuscript received, May 12, 1961.
† Electronics Lab., General Electric Co., Syracuse, N. Y.

[S.B. Akers Jr., *IRE Trans.* **EC-10** (1961), 604–615]

A brief history of majority logic


604 IRE TRANSACTIONS ON ELECTRONIC COMPUTERS

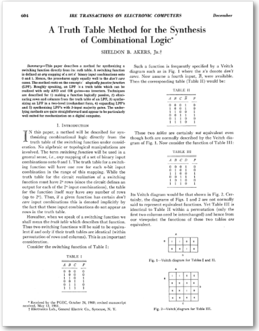
A Truth Table Method for the Synth… of Combinational Logic

SHELDON B. AKERS, Jr.

> Before leaving this section on synthesis, several comments seem appropriate. The reader's first reaction to the foregoing may well be that the one thing which the general area of switching circuit theory does *not* need is *another* method for synthesizing combinational logic. However, this method does offer several features which may make it more desirable in certain applications:

[S.B. Akers Jr., *IRE Trans.* **EC-10** (1961), 604–615]

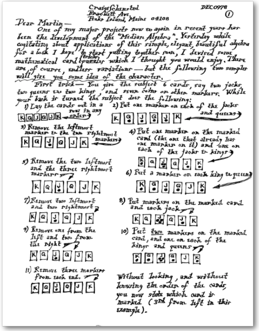
A brief history of majority logic



604 IRE TRANSACTIONS ON ELECTRONIC COMPUTERS December

A Truth Table Method for the Synthesis of Combinational Logic*

SHELDON B. AKERS, JR.†



Craige Schensted
Brackett Ave.
Peaks Island, Maine 04108

DEC 0978

①

Dear Martin —

One of my major projects now & again in recent years has been the development of the "Median Algebra". Yesterday while cogitating about applications of this simple, elegant, beautiful algebra for a book I hope to start putting together soon, I devised some mathematical card tricks which I thought you would enjoy. There are, of course, endless variations — but the following two samples will give you some idea of the character.

First trick — You give the subject 6 cards, say two jacks, two queens & two kings, and seven coins or other markers. While your back is turned the subject does the following:

1) Lay the cards out in a row in any order

2) Put one marker on each of the jacks and queens

3) Remove the leftmost marker & the two rightmost markers

4) Put one marker on the marked card (the one that already has one marker on it) and one on each of the jacks & kings

5) Remove the two leftmost and the three rightmost markers

6) Put a marker on each king & queen

7) Remove two leftmost and two rightmost

8) Put markers on the marked card and each jack

9) Remove one from the left and two from the right

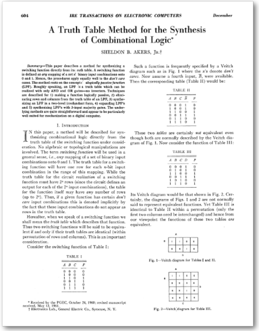
10) Put two markers on the marked card, and one on each of the kings and queens

11) Remove three markers from each end.

Without looking, and without knowing the order of the cards, you now state which card is marked (3rd from left in this example).

[C. Schensted, *Letter to Martin Gardner*, Dec 9, 1978]

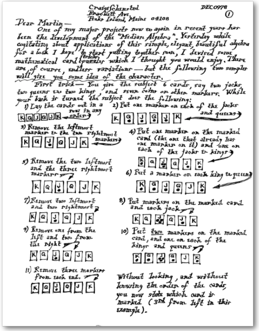
A brief history of majority logic


604 IRE TRANSACTIONS ON ELECTRONIC COMPUTERS December

A Truth Table Method for the Synthesis of Combinational Logic*

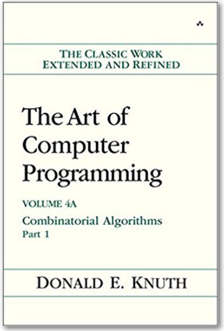
SHELDON B. AKERS, Jr.†

I. Introduction


Craigs Schonstedt
Brackett Ave.
Peaks Island, Maine 04108
DEC 0978

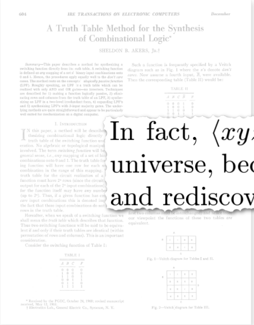
Dear Martin—
One of my major projects now and again in recent years has been the development of the "Median Algebra". Yesterday while cogitating about applications of this simple, elegant, beautiful algebra for a book I hope to start putting together soon, I devised some mathematical card puzzles (tricks) which I thought you would enjoy. There are, of course, endless variations — but the following two samples will give you some idea of the character.

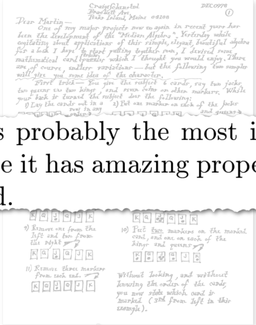
First trick — You give the subject 6 cards, say two jacks, two queens and two kings, and seven coins or other markers. While your back is turned the subject does the following:

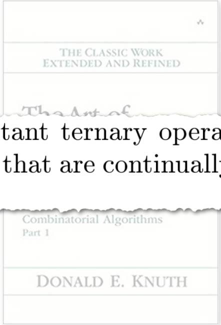


[D.E. Knuth, *The Art of Computer Programming* **4A** (2011)]

A brief history of majority logic



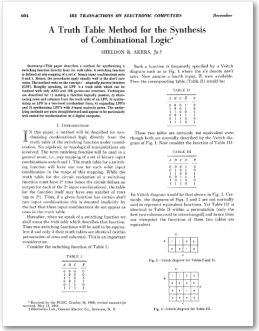




In fact, $\langle xyz \rangle$ is probably the most important ternary operation in the entire universe, because it has amazing properties that are continually being discovered and rediscovered.

[D.E. Knuth, *The Art of Computer Programming* **4A** (2011)]

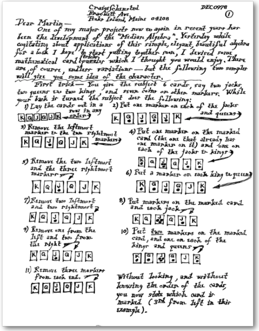
A brief history of majority logic



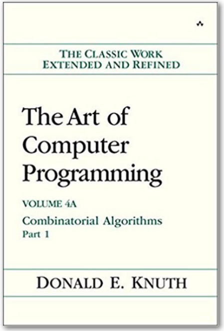
604 IRE TRANSACTIONS ON ELECTRONIC COMPUTERS December

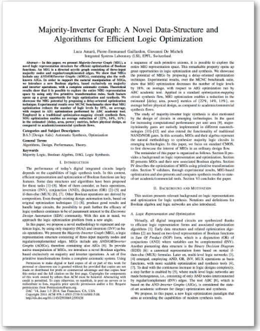
A Truth Table Method for the Synthesis of Combinational Logic*

SHELDON B. AKERS, JR.†



Dear Martin—





Majority-Inverter Graph: A Novel Data-Structure and Algorithms for Efficient Logic Optimization

Luca Amarù, Pierre-Emmanuel Gaillardon, Giovanni De Micheli

Integrated Systems Laboratory (LSI), EPFL, Switzerland

[L.G. Amarù, P.-E. Gaillardon, and G. De Micheli, *DAC* **51** (2014), 194:1–194:6]

Majority function

$\langle x_1 x_2 x_3 \rangle$

Majority function

$$\langle x_1 x_2 x_3 \rangle = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3)$$

Majority function

$$\begin{aligned}\langle x_1x_2x_3\rangle &= (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3)\\ &= x_1x_2 \vee x_1x_3 \vee x_2x_3\end{aligned}$$

Majority function

$$\langle x_1 x_2 x_3 \rangle = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3) \qquad \langle x_1 \ldots x_n \rangle = [x_1 + \cdots + x_n > \tfrac{n}{2}]$$
$$= x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$$

Majority function

$$\langle x_1 x_2 x_3 \rangle = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3)$$
$$= x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$$

$$\langle x_1 \ldots x_n \rangle = [x_1 + \cdots + x_n > \tfrac{n}{2}]$$

Majority rule

$$\langle x_1 x_1 x_2 \rangle = x_1$$
$$\langle x_1 \bar{x}_1 x_2 \rangle = x_2$$

Majority function

$$\langle x_1 x_2 x_3 \rangle = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3)$$
$$= x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$$

$$\langle x_1 \ldots x_n \rangle = [x_1 + \cdots + x_n > \tfrac{n}{2}]$$

Majority rule

$$\langle x_1 x_1 x_2 \rangle = x_1$$
$$\langle x_1 \bar{x}_1 x_2 \rangle = x_2$$

$$\langle x_1 \ldots x_1 x_2 \ldots x_{\lceil \frac{n}{2} \rceil} \rangle = x_1$$
$$\langle x_1 \bar{x}_1 x_2 \ldots x_{n-1} \rangle = \langle x_2 \ldots x_{n-1} \rangle$$

Majority function

$$\langle x_1 x_2 x_3 \rangle = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3)$$
$$= x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$$

$$\langle x_1 \ldots x_n \rangle = [x_1 + \cdots + x_n > \tfrac{n}{2}]$$

Majority rule

$$\langle x_1 x_1 x_2 \rangle = x_1$$
$$\langle x_1 \bar{x}_1 x_2 \rangle = x_2$$

$$\langle x_1 \ldots x_1 x_2 \ldots x_{\lceil \frac{n}{2} \rceil} \rangle = x_1$$
$$\langle x_1 \bar{x}_1 x_2 \ldots x_{n-1} \rangle = \langle x_2 \ldots x_{n-1} \rangle$$

Containment of AND and OR

$$\langle x_1 0 x_2 \rangle = x_1 \wedge x_2$$
$$\langle x_1 1 x_2 \rangle = x_1 \vee x_2$$

Majority function

$$\langle x_1x_2x_3\rangle = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3)$$
$$= x_1x_2 \vee x_1x_3 \vee x_2x_3$$

$$\langle x_1 \ldots x_n\rangle = [x_1 + \cdots + x_n > \tfrac{n}{2}]$$

Majority rule

$$\langle x_1x_1x_2\rangle = x_1$$
$$\langle x_1\bar{x}_1x_2\rangle = x_2$$

$$\langle x_1 \ldots x_1x_2 \ldots x_{\lceil \frac{n}{2}\rceil}\rangle = x_1$$
$$\langle x_1\bar{x}_1x_2 \ldots x_{n-1}\rangle = \langle x_2 \ldots x_{n-1}\rangle$$

Containment of AND and OR

$$\langle x_1 0 x_2\rangle = x_1 \wedge x_2$$
$$\langle x_1 1 x_2\rangle = x_1 \vee x_2$$

$$\langle x_1 \ldots x_{\lceil \frac{n}{2}\rceil} 0 \ldots 0\rangle = x_1 \wedge \cdots \wedge x_{\lceil \frac{n}{2}\rceil}$$
$$\langle x_1 \ldots x_{\lceil \frac{n}{2}\rceil} 1 \ldots 1\rangle = x_1 \vee \cdots \vee x_{\lceil \frac{n}{2}\rceil}$$

Majority: Algebraic rules

Commutativity rule

$$\langle xyz \rangle = \langle yzx \rangle = \langle zxy \rangle$$

Majority: Algebraic rules

Commutativity rule

$\langle xyz \rangle = \langle yzx \rangle = \langle zxy \rangle$

Associativity rule

$\langle xu\langle yuz \rangle \rangle = \langle \langle xuy \rangle uz \rangle$

Mnemonic: $(x \circ (y \circ z)) = ((x \circ y) \circ z)$

Majority: Algebraic rules

Commutativity rule

$\langle xyz \rangle = \langle yzx \rangle = \langle zxy \rangle$

Associativity rule

$\langle xu\langle yuz \rangle\rangle = \langle\langle xuy \rangle uz \rangle$

Mnemonic: $(x \circ (y \circ z)) = ((x \circ y) \circ z)$

Distributivity rule

$\langle xu\langle yvz \rangle\rangle = \langle\langle xuy \rangle v \langle xuz \rangle\rangle$

Mnemonic: $(x \circ (y \times z)) = ((x \circ y) \times (x \circ z))$

Majority: Algebraic rules

Commutativity rule

$\langle xyz \rangle = \langle yzx \rangle = \langle zxy \rangle$

Associativity rule

$\langle xu\langle yuz \rangle\rangle = \langle\langle xuy \rangle uz \rangle$

Mnemonic: $(x \circ (y \circ z)) = ((x \circ y) \circ z)$

Distributivity rule

$\langle xu\langle yvz \rangle\rangle = \langle\langle xuy \rangle v \langle xuz \rangle\rangle$

Mnemonic: $(x \circ (y \times z)) = ((x \circ y) \times (x \circ z))$

Inverter propagation rule

$\langle \bar{x}\bar{y}\bar{z} \rangle = \overline{\langle xyz \rangle}$

Results and motivation from circuit complexity

- NC^1 contains families of Boolean circuits with logarithmic depth, and a polynomial number of 2-input gates, and inverters

Results and motivation from circuit complexity

- NC^1 contains families of Boolean circuits with logarithmic depth, and a polynomial number of 2-input gates, and inverters
- AC^0 contains families of Boolean circuits with constant depth, a polynomial number of AND and OR gates with unbounded fan-in, and inverters

Results and motivation from circuit complexity

- NC^1 contains families of Boolean circuits with logarithmic depth, and a polynomial number of 2-input gates, and inverters
- AC^0 contains families of Boolean circuits with constant depth, a polynomial number of AND and OR gates with unbounded fan-in, and inverters
- TC^0 contains families of Boolean circuits with constant depth, a polynomial number of MAJ gates with unbounded fan-in, and inverters

Results and motivation from circuit complexity

- NC^1 contains families of Boolean circuits with logarithmic depth, and a polynomial number of 2-input gates, and inverters
- AC^0 contains families of Boolean circuits with constant depth, a polynomial number of AND and OR gates with unbounded fan-in, and inverters
- TC^0 contains families of Boolean circuits with constant depth, a polynomial number of MAJ gates with unbounded fan-in, and inverters
- Relationship: $AC^0 \subsetneq TC^0 \subseteq NC^1$

Results and motivation from circuit complexity

- NC^1 contains families of Boolean circuits with logarithmic depth, and a polynomial number of 2-input gates, and inverters
- AC^0 contains families of Boolean circuits with constant depth, a polynomial number of AND and OR gates with unbounded fan-in, and inverters
- TC^0 contains families of Boolean circuits with constant depth, a polynomial number of MAJ gates with unbounded fan-in, and inverters
- Relationship: $AC^0 \subsetneq TC^0 \subseteq NC^1$
- Examples: integer division and integer multiplication are in TC^0, but not in AC^0

Express majority-n in terms of majority-3

One "fascinating" property of AND and OR

Express majority-n in terms of majority-3

One "fascinating" property of AND and OR

$$x_1 \wedge x_2 \wedge \cdots \wedge x_{n-1} \wedge x_n = (x_1 \wedge (x_2 \wedge (\cdots (x_{n-1} \wedge x_n) \cdots)))$$

$$x_1 \vee x_2 \vee \cdots \vee x_{n-1} \vee x_n = (x_1 \vee (x_2 \vee (\cdots (x_{n-1} \vee x_n) \cdots)))$$

Express majority-n in terms of majority-3

One "fascinating" property of AND and OR

$$x_1 \wedge x_2 \wedge \cdots \wedge x_{n-1} \wedge x_n = (x_1 \wedge (x_2 \wedge (\cdots (x_{n-1} \wedge x_n) \cdots)))$$

$$x_1 \vee x_2 \vee \cdots \vee x_{n-1} \vee x_n = (x_1 \vee (x_2 \vee (\cdots (x_{n-1} \vee x_n) \cdots)))$$

Not so easy with majority

Express majority-n in terms of majority-3

One "fascinating" property of AND and OR

$$x_1 \land x_2 \land \cdots \land x_{n-1} \land x_n = (x_1 \land (x_2 \land (\cdots (x_{n-1} \land x_n) \cdots)))$$

$$x_1 \lor x_2 \lor \cdots \lor x_{n-1} \lor x_n = (x_1 \lor (x_2 \lor (\cdots (x_{n-1} \lor x_n) \cdots)))$$

Not so easy with majority

$$\langle x_1 x_2 x_3 x_4 x_5 \rangle = \langle x_1 \langle x_2 x_3 x_4 \rangle \langle x_5 x_4 \langle x_3 x_2 x_1 \rangle \rangle \rangle$$

Express majority-n in terms of majority-3

One "fascinating" property of AND and OR

$$x_1 \wedge x_2 \wedge \cdots \wedge x_{n-1} \wedge x_n = (x_1 \wedge (x_2 \wedge (\cdots (x_{n-1} \wedge x_n) \cdots)))$$

$$x_1 \vee x_2 \vee \cdots \vee x_{n-1} \vee x_n = (x_1 \vee (x_2 \vee (\cdots (x_{n-1} \vee x_n) \cdots)))$$

Not so easy with majority

$$\langle x_1 x_2 x_3 x_4 x_5 \rangle = \langle x_1 \langle x_2 x_3 x_4 \rangle \langle x_5 x_4 \langle x_3 x_2 x_1 \rangle \rangle \rangle$$

$$\langle x_1 x_2 x_3 x_4 x_5 x_6 x_7 \rangle = \langle x_7 \langle x_3 \langle x_4 x_5 x_6 \rangle \langle x_1 x_2 \langle x_4 x_5 x_6 \rangle \rangle \rangle \langle x_6 \langle x_1 x_2 x_3 \rangle \langle x_4 x_5 \langle x_1 x_2 x_3 \rangle \rangle \rangle \rangle$$

Express majority-n in terms of majority-3

One "fascinating" property of AND and OR

$$x_1 \wedge x_2 \wedge \cdots \wedge x_{n-1} \wedge x_n = (x_1 \wedge (x_2 \wedge (\cdots (x_{n-1} \wedge x_n) \cdots)))$$

$$x_1 \vee x_2 \vee \cdots \vee x_{n-1} \vee x_n = (x_1 \vee (x_2 \vee (\cdots (x_{n-1} \vee x_n) \cdots)))$$

Not so easy with majority

$$\langle x_1 x_2 x_3 x_4 x_5 \rangle = \langle x_1 \langle x_2 x_3 x_4 \rangle \langle x_5 x_4 \langle x_3 x_2 x_1 \rangle\rangle\rangle$$

$$\langle x_1 x_2 x_3 x_4 x_5 x_6 x_7 \rangle = \langle x_7 \langle x_3 \langle x_4 x_5 x_6 \rangle \langle x_1 x_2 \langle x_4 x_5 x_6 \rangle\rangle\rangle \langle x_6 \langle x_1 x_2 x_3 \rangle \langle x_4 x_5 \langle x_1 x_2 x_3 \rangle\rangle\rangle\rangle$$

Open problem: What are the optimum majority-3 networks to realize majority-n?

Monotone functions

Montone functions

A Boolean function $f(x_1, \dots, x_n)$ is monotone if $f_{\bar{x}_i} \to f_{x_i}$ for $1 \leqslant i \leqslant n$.

Monotone functions

Montone functions

A Boolean function $f(x_1, \ldots, x_n)$ is monotone if $f_{\bar{x}_i} \to f_{x_i}$ for $1 \leqslant i \leqslant n$.

Schensted decomposition

If $f(x_1, x_2, x_3, \ldots, x_n)$ is monotone, then

$$f(x_1, x_2, x_3, \ldots, x_n) = \langle f(x_1, x_1, x_3, \ldots, x_n) f(x_1, x_2, x_2, \ldots, x_n) f(x_3, x_2, x_3, \ldots, x_n) \rangle$$

- Since majority-n is monotone, we can use Schensted decomposition to map majority-n into majority-3

Monotone functions

Montone functions

A Boolean function $f(x_1, \dots, x_n)$ is monotone if $f_{\bar{x}_i} \to f_{x_i}$ for $1 \leqslant i \leqslant n$.

Schensted decomposition

If $f(x_1, x_2, x_3, \dots, x_n)$ is monotone, then

$$f(x_1, x_2, x_3, \dots, x_n) = \langle f(x_1, x_1, x_3, \dots, x_n) f(x_1, x_2, x_2, \dots, x_n) f(x_3, x_2, x_3, \dots, x_n) \rangle$$

- Since majority-n is monotone, we can use Schensted decomposition to map majority-n into majority-3
- Inner subfunctions remain monotone $\to$ recursive application

Monotone functions

Montone functions

A Boolean function $f(x_1, \ldots, x_n)$ is monotone if $f_{\bar{x}_i} \to f_{x_i}$ for $1 \leqslant i \leqslant n$.

Schensted decomposition

If $f(x_1, x_2, x_3, \ldots, x_n)$ is monotone, then

$$f(x_1, x_2, x_3, \ldots, x_n) = \langle f(x_1, x_1, x_3, \ldots, x_n) f(x_1, x_2, x_2, \ldots, x_n) f(x_3, x_2, x_3, \ldots, x_n) \rangle$$

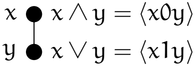
- Since majority-n is monotone, we can use Schensted decomposition to map majority-n into majority-3
- Inner subfunctions remain monotone $\to$ recursive application
- But: Upper bound is exponential!

Majority-n from sorter networks

- Idee: Sort all input bits and pick the middle one from the sorted list

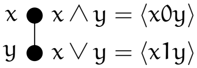
Majority-n from sorter networks

- Idee: Sort all input bits and pick the middle one from the sorted list
- Sorter networks consist only of comparators, which in the Boolean case can be implemented in terms of AND and OR:

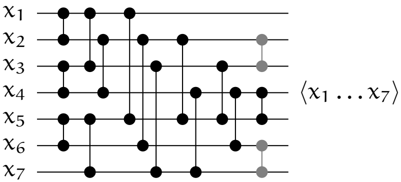


Majority-n from sorter networks

- Idee: Sort all input bits and pick the middle one from the sorted list
- Sorter networks consist only of comparators, which in the Boolean case can be implemented in terms of AND and OR:

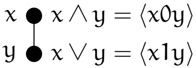


- Example: Sorter networks for 7 bits requires 16 comparisons (optimal), we can drop 2 $\rightarrow$ 28 majority gates

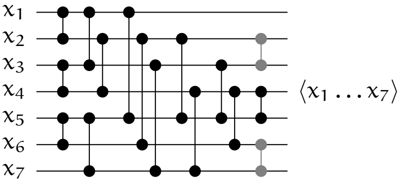


Majority-n from sorter networks

- Idee: Sort all input bits and pick the middle one from the sorted list
- Sorter networks consist only of comparators, which in the Boolean case can be implemented in terms of AND and OR:



- Example: Sorter networks for 7 bits requires 16 comparisons (optimal), we can drop 2 $\rightarrow$ 28 majority gates



Complexity: $O(n \log n)$

Majority-n from median selection

- Median selection: An algorithm that finds the median of given values $\{a_1, \ldots, a_n\}$ using $O(n)$ comparisons (it does not sort *all* elements)

Majority-n from median selection

- Median selection: An algorithm that finds the median of given values $\{a_1, \ldots, a_n\}$ using $O(n)$ comparisons (it does not sort *all* elements)
- $\langle x_1 \ldots x_n \rangle = [\text{median of } \{x_1, \ldots, x_n\}]$

Majority-n from median selection

- Median selection: An algorithm that finds the median of given values $\{a_1, \ldots, a_n\}$ using $O(n)$ comparisons (it does not sort *all* elements)
- $\langle x_1 \ldots x_n \rangle = [\text{median of } \{x_1, \ldots, x_n\}]$
- Good asymptotic upper bound, but the construction is quite complex

Majority-n from median selection

- Median selection: An algorithm that finds the median of given values $\{a_1, \ldots, a_n\}$ using $O(n)$ comparisons (it does not sort *all* elements)
- $\langle x_1 \ldots x_n \rangle = [\text{median of } \{x_1, \ldots, x_n\}]$
- Good asymptotic upper bound, but the construction is quite complex
- Majority-7 based on median selection construction has at least 42 majority gates

Shannon decomposition and majority decomposition

Shannon decomposition

For any Boolean function $f(x_1, \ldots, x_n)$ we have

$$f = x_i \; ? \; f_{x_i} : f_{\bar{x}_i} = x_i f_{x_i} \oplus \bar{x}_i f_{\bar{x}_i}$$

Shannon decomposition and majority decomposition

Shannon decomposition

For any Boolean function $f(x_1, \ldots, x_n)$ we have

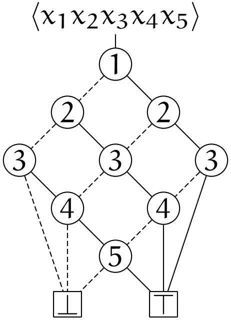
$$f = x_i \,?\, f_{x_i} : f_{\bar{x}_i} = x_i f_{x_i} \oplus \bar{x}_i f_{\bar{x}_i}$$

Majority decomposition [S.B. Akers Jr., 1961]

For a monotone Boolean function $f(x_1, \ldots, x_n)$ we have

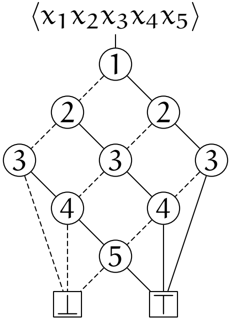
$$f = \langle x_i f_{x_i} f_{\bar{x}_i} \rangle = x_i f_{x_i} \oplus x_i f_{\bar{x}_i} \oplus f_{x_i} f_{\bar{x}_i}$$

From BDDs to majority graphs

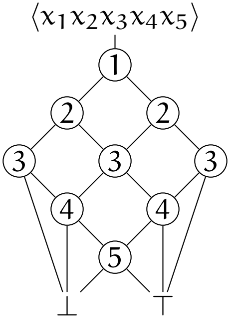


Binary decision diagram

From BDDs to majority graphs

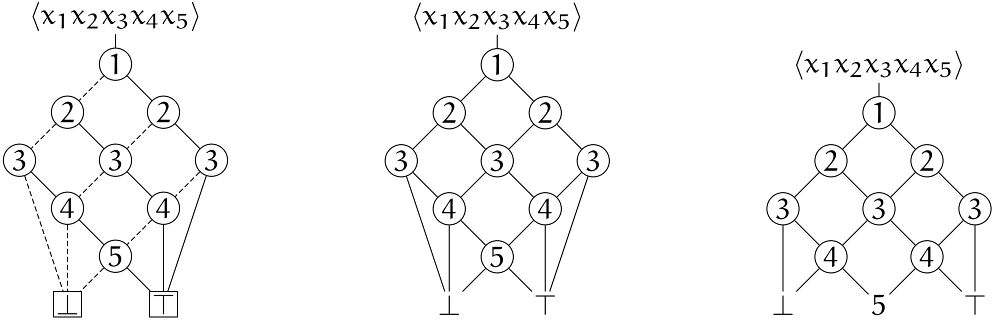


Binary decision diagram



Majority graph

From BDDs to majority graphs



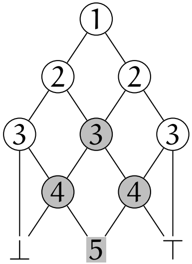
Binary decision diagram   Majority graph   Majority graph (compact)

Upper bounds for majority-n decomposition

n	3	5	7	9	11	13	15	17
Optimum	1	4	7					
BDDs	3	8	15	24	35	48	63	80
Sorter networks	6	18	32	50	70	90	112	142
Median selection*	18	30	42	53	65	77	89	101

*optimistic: takes only into account number of comparators

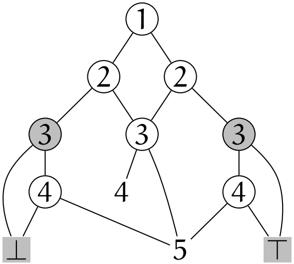
Deriving the optimum majority-5



Apply distributivity rule

$$\langle\langle x_4 x_5 0\rangle x_3 \langle x_4 x_5 1\rangle\rangle = \langle x_4 x_5 \langle 0 x_3 1\rangle\rangle = \langle x_4 x_3 x_5\rangle$$

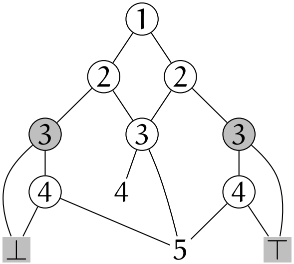
Deriving the optimum majority-5



Apply relevance rule

$$\langle xyz \rangle = \langle xyz_{x/\bar{y}} \rangle$$

Deriving the optimum majority-5

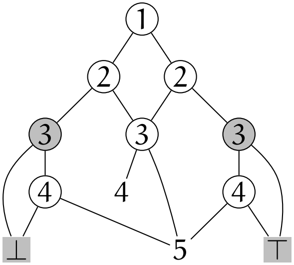


Apply relevance rule

$$\langle xyz\rangle = \langle xyz_{x/\bar{y}}\rangle$$

$$\langle 0x_3\langle 0x_4x_5\rangle\rangle = \langle 0x_3\langle 0x_4x_5\rangle_{0/\bar{x}_3}\rangle = \langle 0x_3\langle \bar{x}_3x_4x_5\rangle\rangle$$

Deriving the optimum majority-5



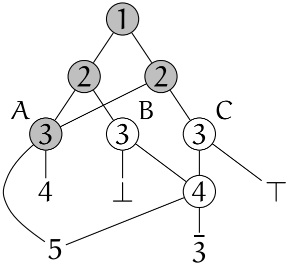
Apply relevance rule

$$\langle xyz \rangle = \langle xyz_{x/\bar{y}} \rangle$$

$$\langle 0x_3 \langle 0x_4x_5 \rangle \rangle = \langle 0x_3 \langle 0x_4x_5 \rangle_{0/\bar{x}_3} \rangle = \langle 0x_3 \langle \bar{x}_3x_4x_5 \rangle \rangle$$

$$\langle 1x_3 \langle 1x_4x_5 \rangle \rangle = \langle 1x_3 \langle 1x_4x_5 \rangle_{1/\bar{x}_3} \rangle = \langle 1x_3 \langle \bar{x}_3x_4x_5 \rangle \rangle$$

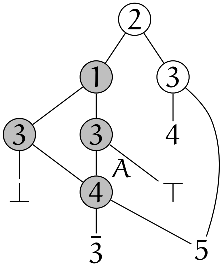
Deriving the optimum majority-5



Apply distributivity rule

$$\langle\langle x_2 A B\rangle x_1 \langle x_2 A C\rangle\rangle = \langle x_2 A \langle B x_1 C\rangle\rangle$$

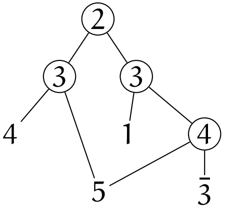
Deriving the optimum majority-5



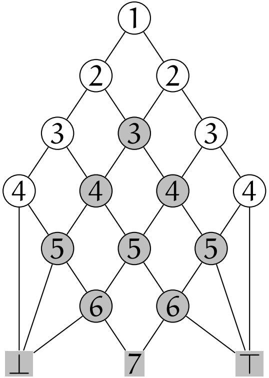
Apply distributivity rule

$$\langle\langle x_3 A 0\rangle x_1 \langle x_3 A 1\rangle\rangle = \langle x_3 A \langle 0 x_1 1\rangle\rangle = \langle x_3 A x_1\rangle$$

Deriving the optimum majority-5



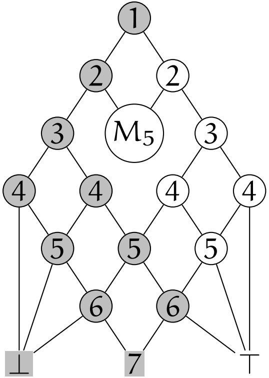
Deriving the optimum majority-7



Identify majority-5

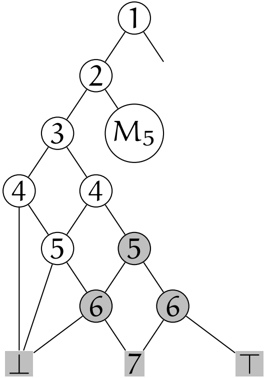
There are actually four majority-5 subnetworks in the graph

Deriving the optimum majority-7



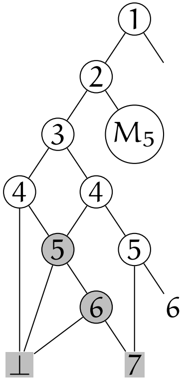
Consider left branch

Deriving the optimum majority-7



Identify majority-3

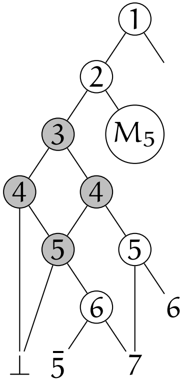
Deriving the optimum majority-7



Relevance

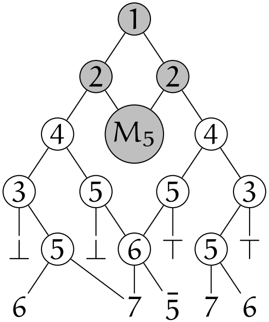
Changes constants into primary inputs

Deriving the optimum majority-7



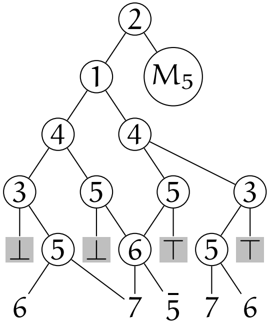
Distributivity

Deriving the optimum majority-7



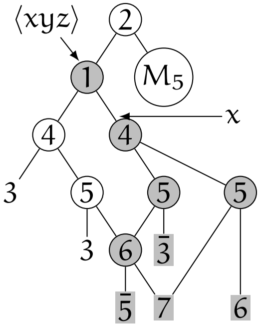
Distributivity

Deriving the optimum majority-7



Remove ⊥ and ⊤

Deriving the optimum majority-7



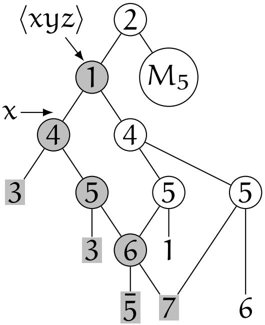
Replacement rule

We have

$$\langle xyz \rangle = \langle wyz \rangle$$

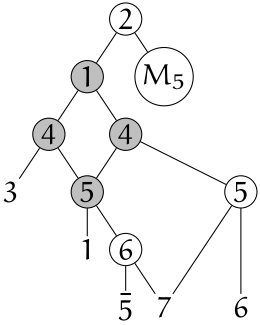
if and only if $(y \oplus z)(w \oplus x) = 0$.

Deriving the optimum majority-7



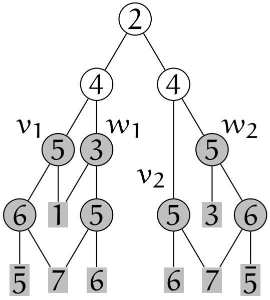
Replacement rule

Deriving the optimum majority-7



Distributivity + M_5 optimum

Deriving the optimum majority-7



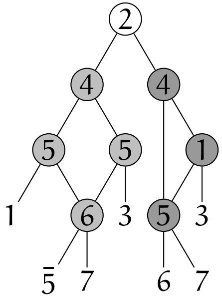
Swapping rule

Let v_1, v_2, w_1, w_2 not depend on x and y. We have

$$\langle x \langle y v_1 w_1 \rangle \langle y v_2 w_2 \rangle \rangle = \langle x \langle y v_2 w_1 \rangle \langle y v_1 w_2 \rangle \rangle,$$

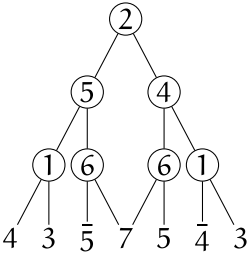
if $(v_1 \oplus v_2)(w_1 \oplus w_2) = 0$.

Deriving the optimum majority-7



Distributivity and relevance

Deriving the optimum majority-7



Optimum result

Conclusions

- Research question: How many majority-3 operations do we need to realize majority-n (precisely)?

Conclusions

- Research question: How many majority-3 operations do we need to realize majority-n (precisely)?
- Constructions that were used to show good asymptotic upper bounds are not helpful for small n

Conclusions

- Research question: How many majority-3 operations do we need to realize majority-n (precisely)?
- Constructions that were used to show good asymptotic upper bounds are not helpful for small n
- Proposed construction method based on BDDs by exploiting decomposition property for monotone functions

Conclusions

- Research question: How many majority-3 operations do we need to realize majority-n (precisely)?
- Constructions that were used to show good asymptotic upper bounds are not helpful for small n
- Proposed construction method based on BDDs by exploiting decomposition property for monotone functions
- Next: Majority-9 and more insight into analytical derivations

The fascinating properties of majority

Mathias Soeken

Integrated Systems Laboratory, EPFL, Switzerland

mathias.soeken@epfl.ch msoeken.github.io msoeken/cirkit