TERNARY AND QUATERNARY LATTICE DIAGRAMS FOR LINEARLY-INDEPENDENT LOGIC, MULTIPLE-VALUED LOGIC, AND ANALOG SYNTHESIS

Marek A. Perkowski, Edmund Pierzchala, and Rolf Drechsler +,

Dept. Electr. Engn., Portland State University, Portland, USA

+ Inst. Comp. Sci., Albert-Ludwigs-University,
Freiburg in Breisgau, Germany

PLAN

- Introduction - layout-driven synthesis.
- Expansions and expansion nodes.
- Max-type versus LI-type lattices.
- Binary LI-type lattices.
- Ternary lattices.
- Quaternary lattices.
- Butterfly algorithm to find best expansions.
- Applications to Fuzzy and analog circuits.

LATTICE DIAGRAMS.

- Review **Binary Lattice Diagrams**.
- Introduce **Ternary and Quaternary Lattice Diagrams**.
- Such diagrams are applicable to **submicron design** and designing new fine-grain digital, analog and mixed FPGAs.
- Diagrams presented here expand the ideas of **Lattice diagrams** (Perkowski, Jeske) and **Linearly Independent** (LI) Logic (Perkowski, Falkowski, Beyl, Sarabi).

THE GOAL OF LATTICE DIAGRAMS

- The goal of **Lattice Diagrams** is **layout-driven logic synthesis** in cellular structures with mostly local connections.
- The concept of a lattice diagram involves three components:

(1) **expansion** of a function (the function corresponds to the initial node in the lattice), which creates several successor nodes of this node,

(2) **joining** of several (not necessarily tautologic) nodes of a tree level to a single node, which is in a sense a reverse operation to the expansion,

(3) a **regular geometry** to which the nodes are mapped, this geometry guides which nodes of the level are to be joined.

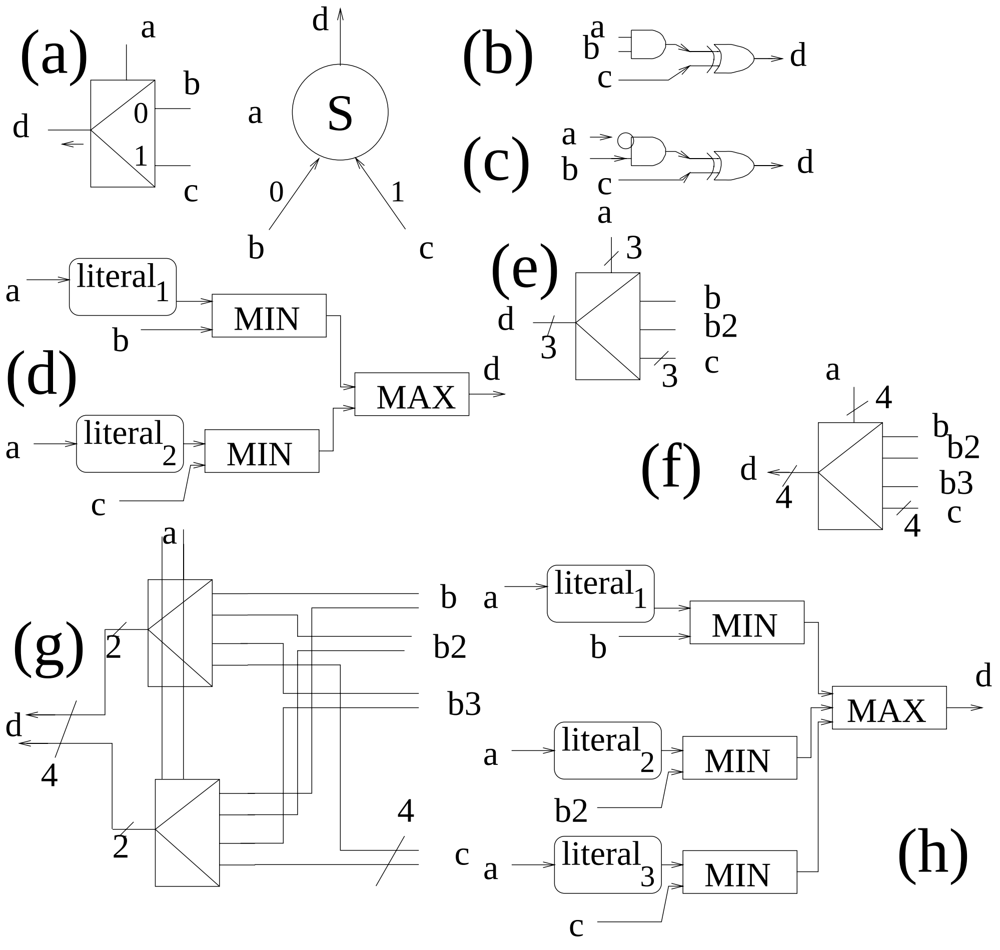
REGULAR LAYOUT GEOMETRY FROM LATTICE DIAGRAMS

- In a regular layout, every cell is connected to 4 (binary lattice), 6 (ternary lattice) or 8 (quaternary lattice) **neighbors** and to a number of vertical, horizontal and diagonal **buses.**
- Cell with n inputs and m outputs is said to have n x m **connectivity pattern**.
- **Ternary lattices** have 3 inputs and 3 outputs from a node.
- **Quaternary lattices** have 4x4 connectivity pattern, it means, 4 inputs and 4 outputs from a node.

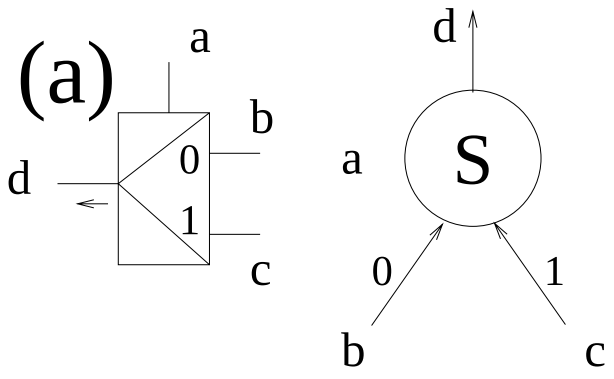
REGULAR LAYOUT GEOMETRY FROM LATTICE DIAGRAMS. II

- Expansions are: Shannon, Davio, nonsingular, fuzzy and analog.
- For each **type of expansion** on nodes, there exists **type of joining operation** for nodes.
- The procedure of building the lattice diagram, i.e. the layout of a function, consist in **expanding** and **joining** nodes in levels iteratively for **(repeated) variables** until all node functions become variables or constants.

EXPANSION NODES FOR BINARY, MULTI-VALUED AND FUZZY FUNCTIONS



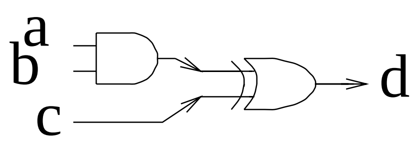
SHANNON EXPANSION NODES.



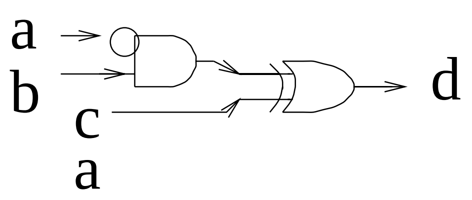
- Shannon (S) expansion: a multiplexer, and a general notation of a 2x2 cell in a Lattice.
- When input a is inverted, the so-called **Reversed Shannon (S') expansion** is executed, which means that the role of inputs b and c is reversed.

DAVIO EXPANSION NODES.

(b)

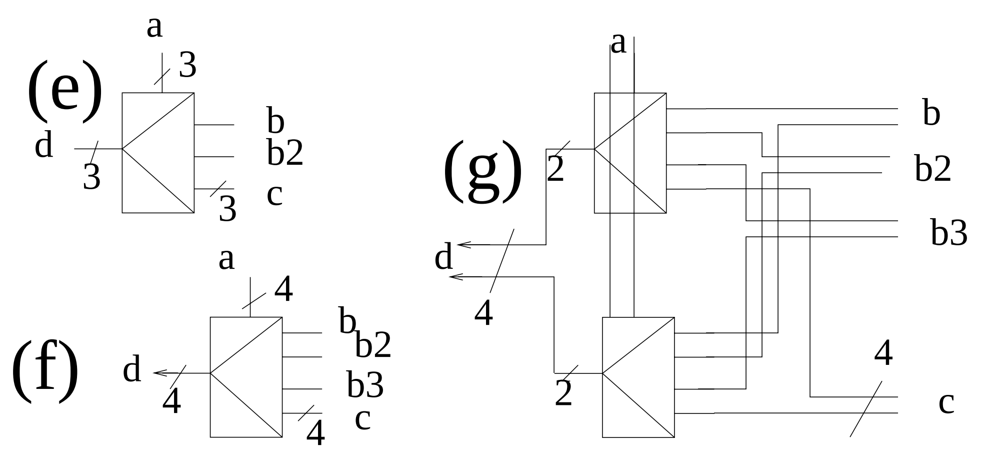


(c)



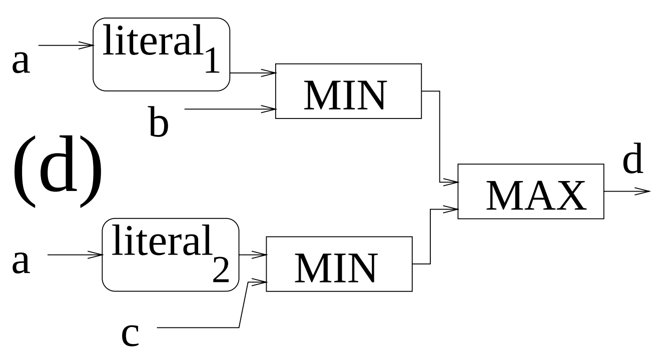
- (b) shows the positive Davio expansion node (pD), and (c) the negative Davio node (nD).
- Such nodes are used in Positive-Polarity, Fixed-Polarity, Kronecker and Pseudo-Kronecker Lattices and their generalizations.

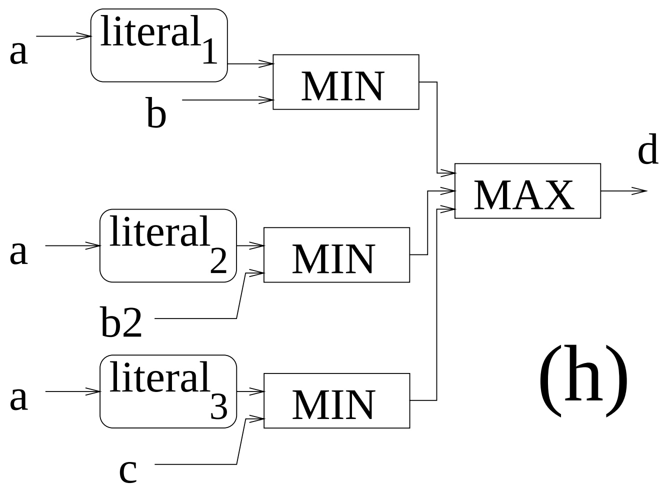
MULTI-VALUED EXPANSION NODES.



- (e) presents Shannon node for ternary logic, (f) Shannon node for quaternary logic, and (g) realization of the quaternary Shannon node from (f) in binary logic.
- Two binary signals routed together simulate a 4-valued signal.

FUZZY LOGIC EXPANSION NODES.





(d) DFL (Disjoint Fuzzy Logic) with 2 literals.

(h) DFL with 3 literals.

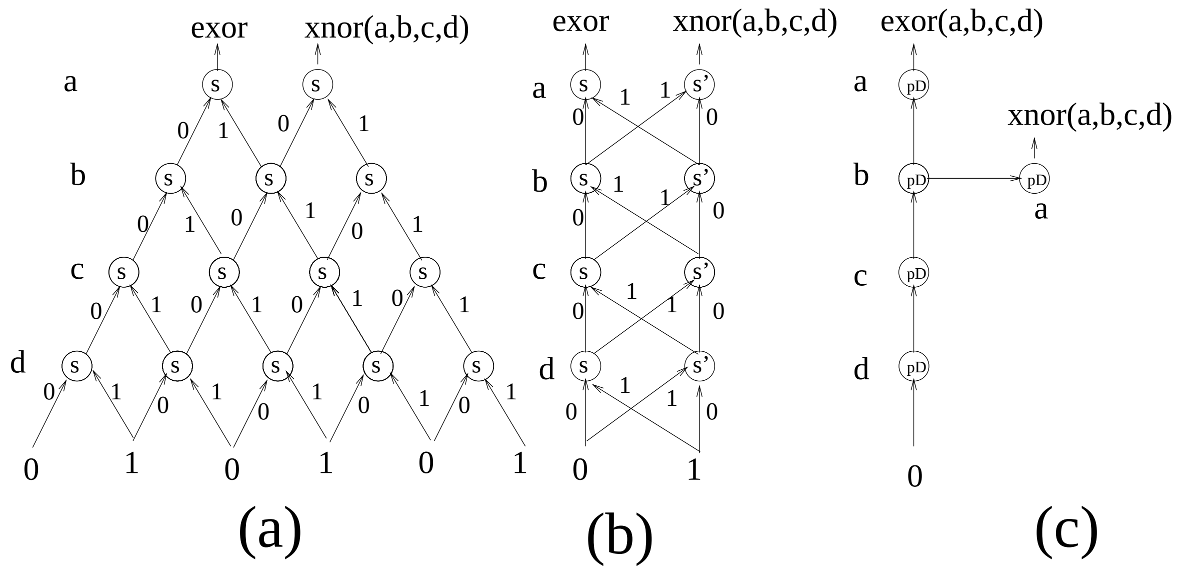
EXISTENCE OF JOINING OPERATIONS AS A CONDITION OF BUILDING LATTICES

- We denote max-type operations by $+$, min-type operations by $\cdot$.
- It can be observed, that a fundamental condition for **existence of joining operations** is that in the underlying algebraic structure any two literals are disjoint.
- In binary, this property reduces to $a \cdot \bar{a} = 0$.
- Existence of joining operations is the condition of being able to create lattice diagrams.
- This condition leads to binary and multiple-valued (MV) Max-type lattices.
- The principle of operation of binary max-type lattices is that any path in a diagram that includes x and $\bar{x}$ **cancells**.

EXISTENCE OF JOINING OPERATIONS FOR LI-TYPE LOGIC

- EXOR function is: $a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$.
- Thus, $a \oplus a = a\bar{a} + \bar{a}a = 0$.
- This leads to Linearly-Independent type (LI) lattices.
- The principle of operation of LI-type lattices is that any two identical paths to the root in the diagram cancel one another ($x \oplus x = 0$).

COMPARISON OF THREE TYPES OF LATTICES FOR TWO-OUTPUT EXOR/XNOR FUNCTION



(a) 2x2 lattice with S, (b) 3x3 lattice with S and S',

(c) 2x2 lattice with pD and pD'.

CREATION OF A POSITIVE DAVIO LEVEL IN A LATTICE

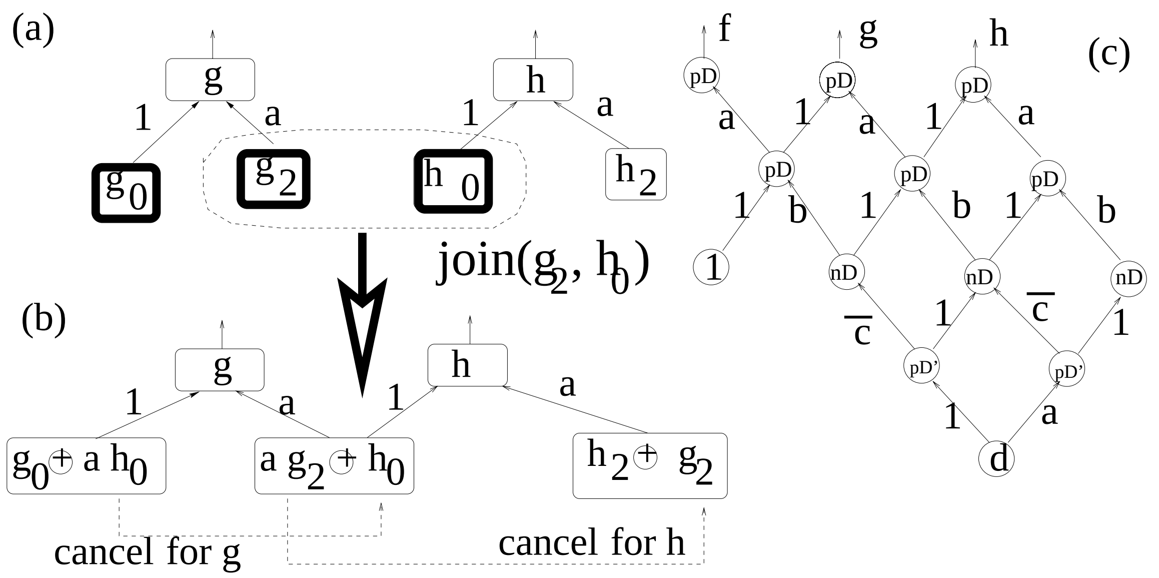


Figure 1: *(a) two expanded nodes before joining, (b) layer of lattice after joining operation on nodes* g_2 *and* h_0*, (c) Fixed-Polarity RM Lattice for functions* f, g, h*.*

BINARY LI-TYPE LATTICES FOR SYMMETRIC AND NON-SYMMETRIC FUNCTIONS

- When a function is symmetric, variables are not repeated.
- Figures clearly demonstrate an advantage of having higher connection patterns and more general expansion types.
- Predictability and equality of delays should be appreciated in all lattices.
- But what about lattice realization of **non-symmetric functions**?
  - **Polarized Pseudo-Kronecker** symmetries (Drucker/Perkowski) are much more general than known symmetries of functions. Using them, more functions can be realized without repeating variables.
  - functions that do not have the Polarized Pseudo-Kronecker symmetries can be still realized in lattices with **repeated variables** (Perkowski/Jeske).

JOINING OPERATION FOR LI-TYPE LATTICES

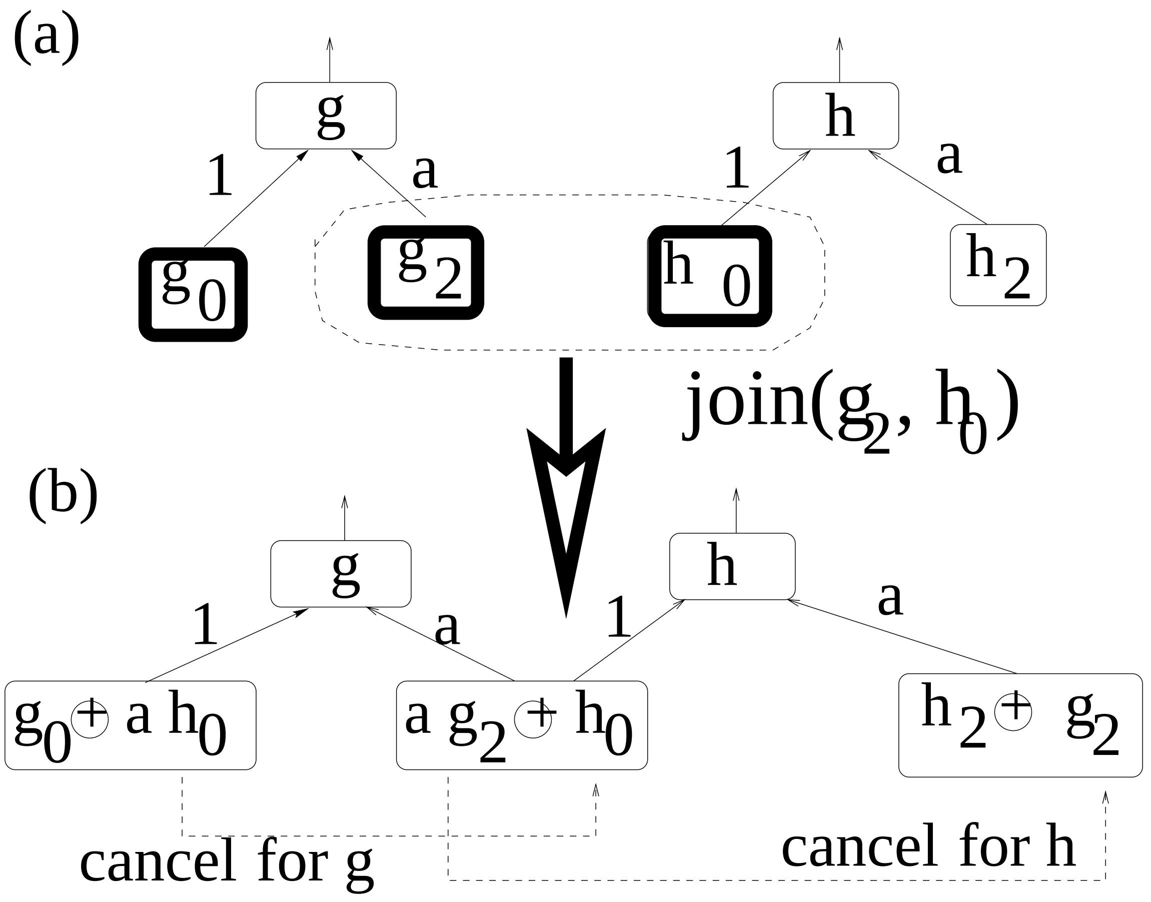


Figure 2: *(a) two expanded nodes before joining, (b) layer of lattice after joining operation on nodes g_2 and h_0.*

JOINING OPERATION FOR LI-TYPE LATTICES. II

- Although shown here only for pD nodes and an ordered lattice, the same principle is used for more complex expansions and lattice diagrams of the LI type.
- The joining rule is: $g_2 \; JOIN \; h_0 = ag_2 \oplus h_0$, which means that nodes representing functions $g_2 = g_0 \oplus g_1$ and h_0 are joined together to create a new node with function $ag_2 \oplus h_0$.
- The **correction terms** ah_0 and ag_2 are propagated to left and right, respectively.

CREATING A DIAGRAM BY EXPANDING AND JOINING OPERATIONS. I

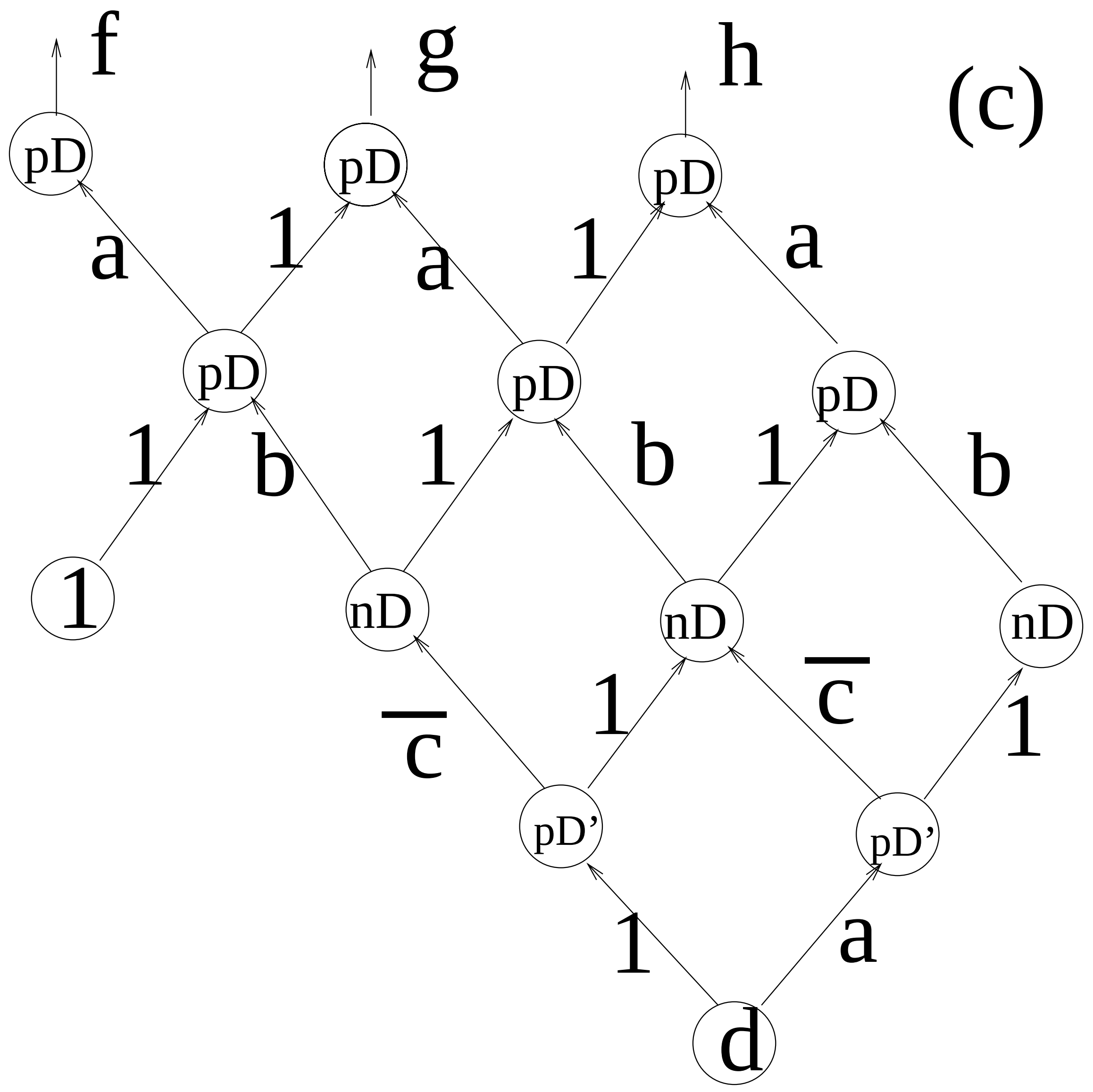


Figure 3: *Fixed-Polarity RM Lattice for functions f, g, h.*

CREATING A DIAGRAM BY EXPANDING AND JOINING OPERATIONS. II

- Fixed-Polarity Reed-Muller Lattice Diagram (expansions pD and nD) for functions:

$$f = a \oplus ab\bar{c}d,$$
$$g = 1 \oplus b\bar{c}d \oplus a\bar{c}d \oplus abd \oplus ab\bar{c}d,$$
$$h = \bar{c}d \oplus bd \oplus ab\bar{c}d \oplus a\bar{c}d \oplus abd \oplus ad.$$

- Variable a is repeated once more in the bottom level of the lattice.
- The expansion in this level is pD', which means, a reversed pD, that is a pD expansion with reversed role of data inputs.

CREATING A DIAGRAM BY EXPANDING AND JOINING OPERATIONS. III

- In some types of expansions the propagation of correction terms is **only to right,** or only to left.
- In some other expansions, especially the non-canonical ones, **more powerful corrections types** are created, and the algorithm selects the correction rule evaluated as the one leading to the simplest next level of the lattice.
- **Selecting** the order of **(repeated) variables** and the **expansion type** in each node are the most important and difficult problems to be solved.

TERNARY AND QUATERNARY LATTICES.

- Binary Shannon expansions can be easily generalized to 3-valued and 4-valued Shannon expansions.
- Lattices for them require 3 inputs and 3 outputs from a node, and 4 inputs and 4 outputs from a node, respectively.
- 3- and 4- valued counterparts of S' are created.
- Ternary and quaternary lattices can be created using corresponding "expansion" and "join" formulas.
- This way, Post-type and Galois-type lattices are created in an uniform way.

TERNARY AND QUATERNARY LATTICES. II

- However, the two kinds of principles, of **creating the expansion** and of the **joining rules**, remain the same: disjoint literals for max-type lattices, and $a + (-a) = 0$ term cancelling for LI lattices (which generalizes the rule $a \oplus a = 0$ of Galois Field (2) given earlier).
- The lattices have advantages especially for (nearly) symmetric functions and strongly unspecified functions that can be completed to symmetric functions.

REGULAR LAYOUT

- By a **regular layout** we understand a layout of indentical cells that connect by **abutting**.
- By a **complete layout structure** we understand connection pattern between cells, that allows to realize every symmetric function without repeating variables.
- It can be proved that in a 2x2 lattice **every** binary symmetric function can be realized without variable repetitions, and with connections between cells having the same length.
- Thus, **lattice layout for binary logic is regular and complete**.

WHEN REGULAR LAYOUT CAN BE CREATED.

- In contrast to binary functions, symmetric **ternary functions cannot be realized in regular 2-dimensional 3x3 lattices**.
- Although we created 3x3 lattices that can realize every symmetric ternary function without variable repetitions, it is not possible to find regular layouts for realizing them.
- Thus the **cells distances** in subsequent levels **grow**.
- Hopefully, it is not a practical problem for small functions realized in MV logic, but the beautiful simplicity of binary realizations does not longer exist.

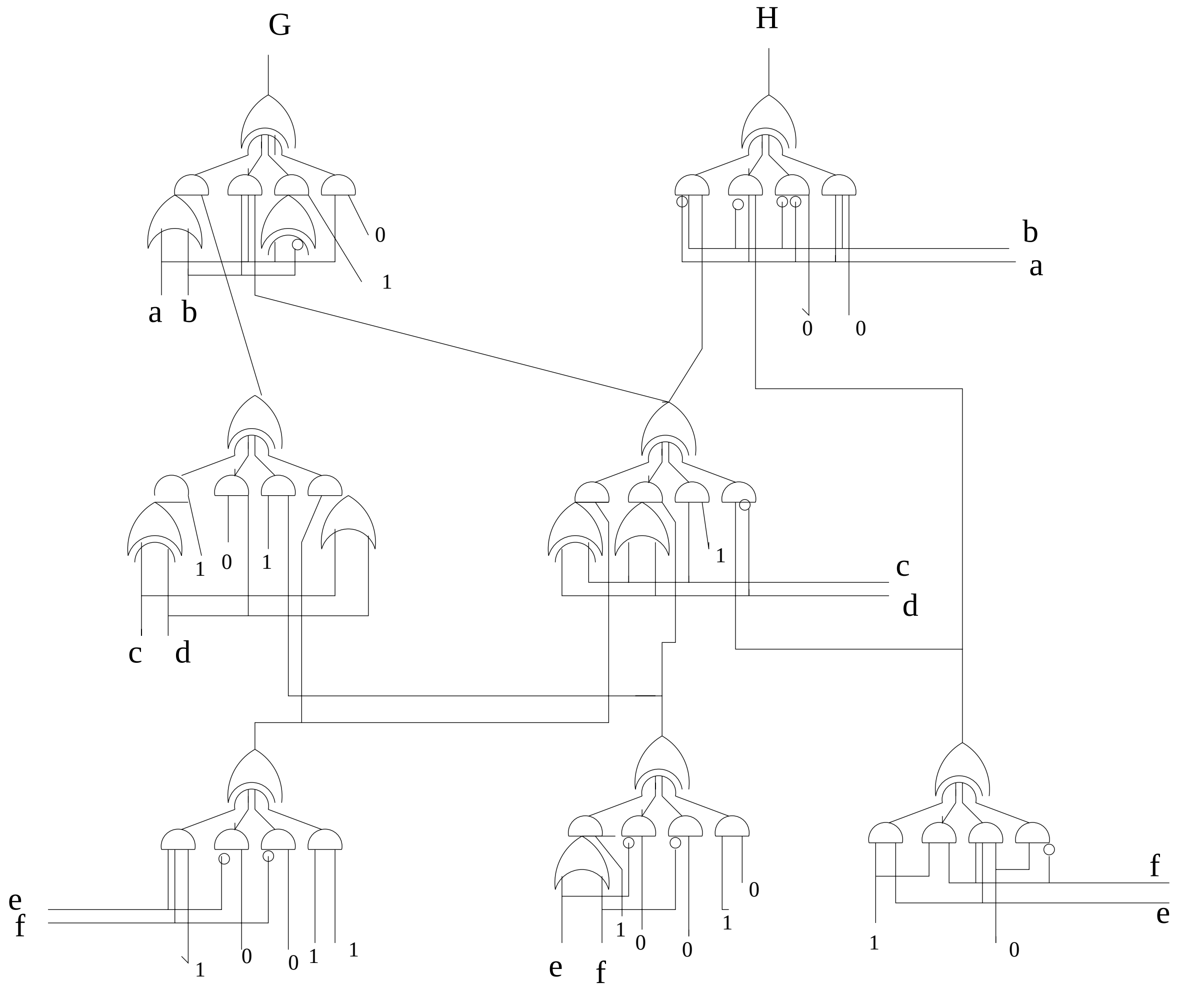
WHEN REGULAR LAYOUT CAN BE CREATED. II

- Thus, if mapped to a 2-dimensional space, the ternary lattices **are either regular and not complete,** or **complete but not regular**.
- It is still possible to obtain regular and complete 3x3 lattices assuming layout of cells in a three-dimensional space.
- But it is not possible to create regular layout for 4x4 lattices, because our Universe is 3-dimensional.

QUATERNARY LI LATTICES.

- As shown before, **pairs of binary variables** correspond to **4-valued variables**.
- Although here we discuss LI lattices for only two variables in each **variable block**, all concepts and algorithms can be expanded to variable blocks of arbitrary size.
- Next Figure shows an example of a circuit obtained by substituting nodes of a quaternary LI lattice diagram with their circuits.
- The LI Lattice diagrams for pairs of variables are created similarly to lattices for single variables.
- Nodes are now for pairs of variables, and nonsingular expansions of LI logic are used.
- Every node has at most 4 inputs.

LI PSEUDO-KRONECKER DECISION LATTICE DIAGRAM FOR VARIABLE BLOCKS {a,b},{c,d},{e,f} to function H, G.


G
H
0
1
a
b
b
a
0
0
1
0
1
c
d
1
c
d
e
f
1
0
0
1
1
1
0
0
1
0
e
f
f
e
1
0

QUATERNARY LI LATTICES. II

- Instead selecting among only three expansions, S, pD and nD, the choice in every level of nodes is among all 840 nonsingular expansions in exact algorithm.
- This is the maximum number of nonsingular expansions for a pair of variables
- Or, some subset of the expansions.
- The same type of expansion is selected in Kronecker type lattices.
- Various expansions are selected in nodes of Pseudo-Kronecker type lattices.
- The joinings are based on the same principles as before.

QUATERNARY LI LATTICES. III

- The lattices for all single outputs of a multi-output function are created together, level-by-level from their root nodes (outputs).
- In every level, the possible expansions are evaluated based on the complexity of the next level (look-ahead strategy).
- The best expansion found by the Polarity Selecting Algorithm for a level is next applied to all nodes (Kronecker types) from the level of the multi-output diagram.
- In Pseudo type of lattices, the expansion decision for each node is done separately.
- The algorithm below is used for small functions, approximate algorithms for larger functions.

BUTTERFLY DIAGRAMS TO FIND BEST LI EXPANSIONS

- **Butterfly diagrams** in Reed-Muller Logic allow to create all fixed polarity expansions by transforming from polarity to polarity.
- They do this just by incremental exoring of some terms from the forms.
- This way, all forms of certain type are systematically created without even creating their **expansion matrices** M and without calculating their inverse matrices M^{-1}.
- The concept of Gray-code ordering of all Generalized Reed-Muller polarities was applied to find the exact minimum GRM form (Zeng/Perkowski).
- Similar ideas proposed here for the LI forms.

PROPERTIES FOR BUTTERFLY DIAGRAM ALGORITHMS

Property 1. The following rule **BR** holds

$$f_1(x_1,x_2)SF_2(x_3,...,x_n) \oplus f_3(x_1,x_2)SF_4(x_3,...,x_n) =$$

$$[f_1(x_1,x_2) \oplus f_3(x_1,x_2)]SF_2(x_3,...,x_n)$$

$$\oplus f_3(x_1,x_2)[SF_2(x_3,...,x_n) \oplus SF_4(x_3,...,x_n)]$$

where $f_1(x_1,x_2)$ and $f_3(x_1,x_2)$ are arbitrary LI functions, and $SF_2(x_3,...,x_n)$ and $SF_4(x_3,...,x_n)$ are the corresponding to them data input (DI) functions.

PROPERTIES FOR BUTTERFLY DIAGRAM ALGORITHMS. II

Property 2. Any nonsingular expansion can be obtained by a repeated application of Rule BR to pairs of functions

$[\ f_1(x_1, x_2), SF_2(x_3, .., x_n)],[f_3(x_1, x_2), SF_4(x_3, .., x_n)]$.

This way, rule BR describes simultaneous EXOR-ing of columns in matrix M and corresponding columns in M^{-1}.

But how to select the pairs of functions?

Property 3. In matrix M, as well as in matrix M^{-1}, any column can be replaced by a linear combination of itself with other columns.

Thus, any *polarity expansion* can be obtained by a repeated application of the basic rule BR to certain selected columns.

BUTTERFLY DIAGRAMS TO FIND BEST EXPANSIONS

- In general, there is no recursive way to define the universal Butterfly-like diagram for **arbitrary** LI matrix.
- A specific diagram can be once created for a set of variables with **certain number** of elements and for any set of expansion polarities.
- This diagram can be stored in memory, and next used for evaluations for each particular function of the respective number of variables.
- We will call this a **"pre-computed" Butterfly diagram.**

BUTTERFLY DIAGRAMS TO FIND BEST EXPANSIONS

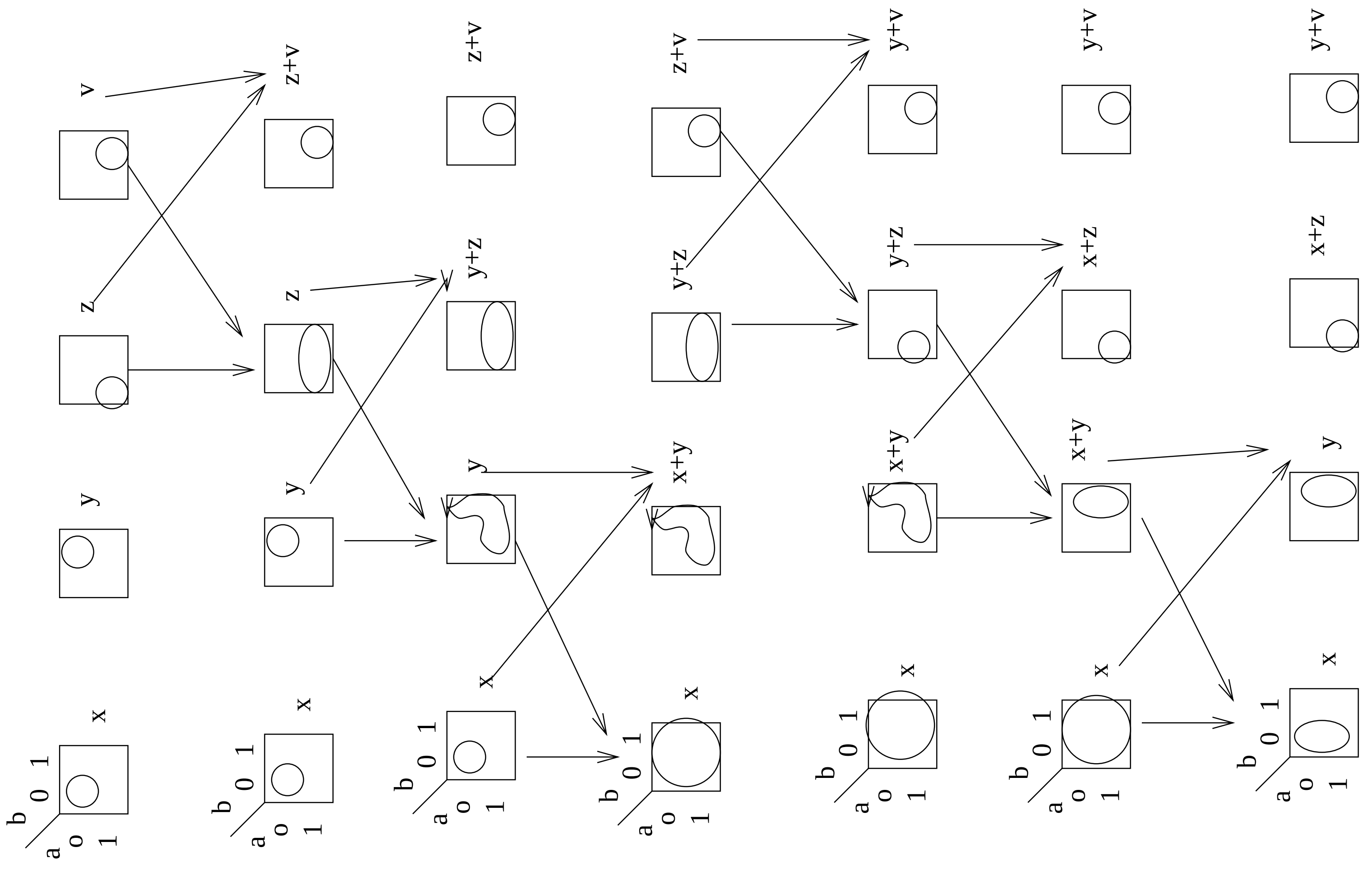
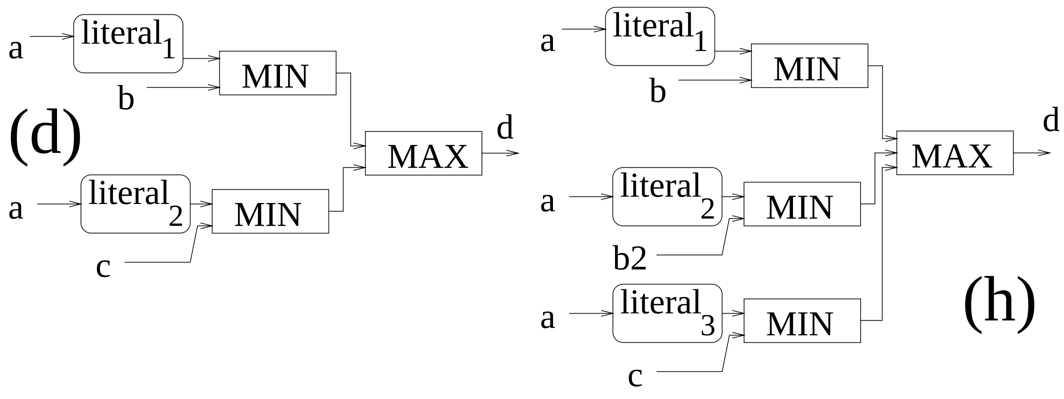


Figure 4: *First part of the Butterfly Diagram to find the best nonsingular expansion by creating expansions for all LI functions of a,b.*

LATTICES FOR FUZZY LOGIC.

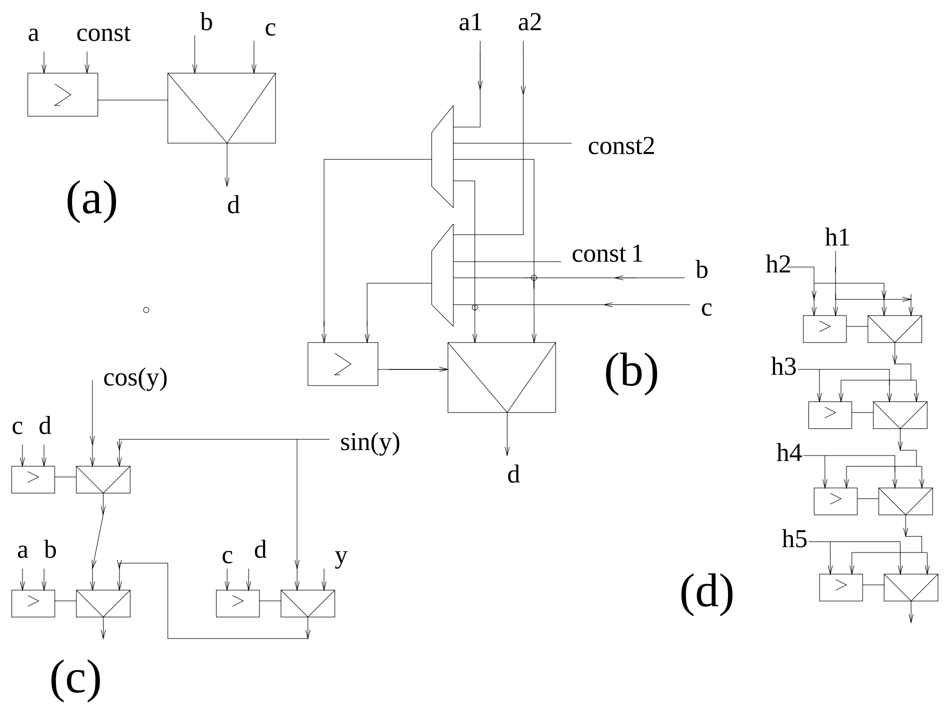
- Because in **standard fuzzy logic** $a \cdot \bar{a} \neq 0$, and $a \oplus a = a\bar{a} + \bar{a}a \neq 0$, both the Max-type and LI methods would not work for it.
- One can define a **negation-less fuzzy logic**, which we call a **Disjoint Fuzzy Logic (DFL)**, in which all fuzzy logic axioms besides those related to negation are satisfied, and negation is simulated by using special type of literals.
- In DFL logic, any two literals $literal_i$, $literal_j$ can have arbitrary shapes, but **must be disjoint;** for any value of $x \in [0, 1]$ $literal_i(x) \cdot literal_j(x) = 0$.

LATTICES FOR FUZZY LOGIC.

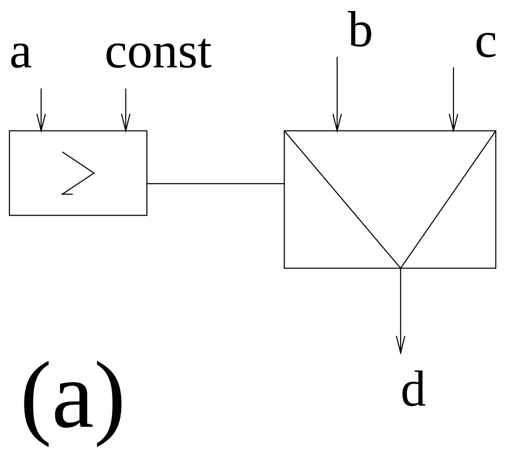


- The literals $literal_1$, $literal_2$, $literal_3$ are all mutually disjoint.
- DFL expansions are realized in **ternary fuzzy lattices**, similar to MV ternary lattices.
- Because of disjoint literals, joining operation can be always performed.

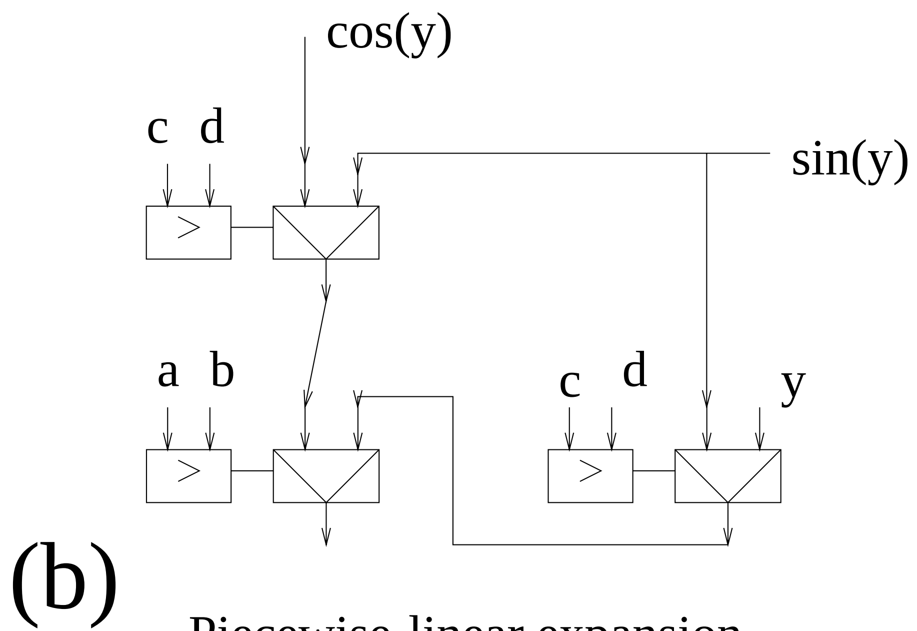
REALIZATION OF ANALOG FUNCTIONS



REALIZATION OF ANALOG FUNCTIONS I.

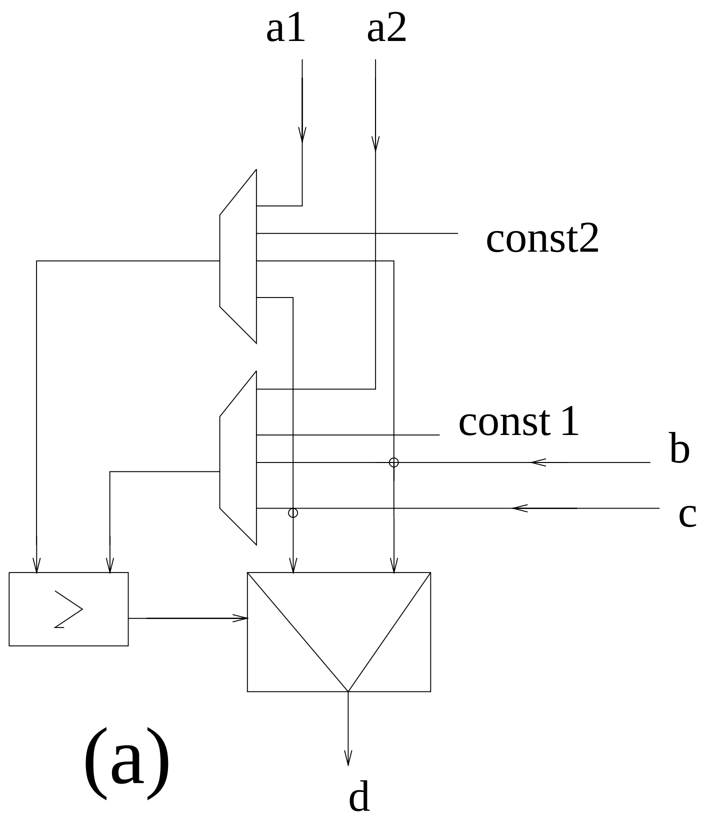


expansion node
for analog logic



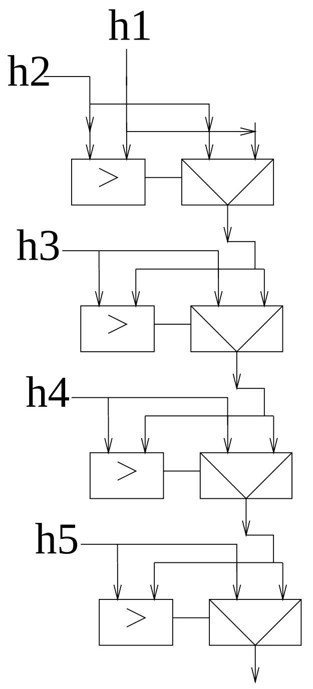
Piecewise-linear expansion
of a continuous function
in a 2x2 - type regular layout

REALIZATION OF ANALOG FUNCTIONS II.



(a)

expansion cell for sorting applications



(b)

Regular 2x2 layout
for max(h1,h2,h3,...h5)

ITERATIVE CIRCUITS AND ANALOG FPGAS.

- Hierarchical design of **iterative** one- and two-dimensional structures.
- Cellular connections of logic blocks, each block realized as a multi-output lattice.
- Created also for discrete circuits with memory.
- Analog counterparts use sample-hold analog memories, which play the same role as flip-flops in discrete technologies.
- Lattices allow thus the realization of cellular memory-less functions, finite state machines, and infinite state machines; realized in analog, binary, or multivalued logic.
- Digital and analog: filters, pipelined image processors, or systolic processors.
- An elliptic ladder filter was mapped to this structure (Pierzchala).
- Rank and median filters, cellular neural nets, equation solvers, and (analog and digital) image processing circuits.

CONCLUSION.

- Presented methods allow for **layout-driven synthesis approaches** to binary, multivalued, linearly-independent, Galois, fuzzy, analog and mixed functions.
- They unify many known expansions, decision diagrams, regular layout geometries and FPGA/FPAA structures.
- Of special interest to **various new technologies** based on **regularity** and **locality** of connections:
  - deep sub-micron technology,
  - binary and MV pass-transistor designs,
  - quantum logic devices,
  - OTA circuits,
  - new fine grain digital and analog FPGAs.

CONCLUSION. (cont.)

- Ternary and quaternary lattices for binary, multi-valued, DFL and analog logic.
- Such diagrams are the **most general lattice diagrams** introduced so far.
- Algorithm for creation of quaternary lattices for binary LI logic.
- Methods applicable to completely specified and **incompletely specified** functions; single-, and **multi-output**.
- Kronecker-like and Pseudo-Kronecker-like generalizations.