LAYOUT-DRIVEN SYNTHESIS FOR SUBMICRON TECHNOLOGY: MAPPING EXPANSIONS TO REGULAR LATTICES

Marek A. Perkowski, Edmund Pierzchala, and Rolf Drechsler +,
Dept. Electr. Engn., Portland State University, Portland, OR 97207
+ Inst. Comp. Sci., Albert-Ludwigs-University, 79110 Freiburg in Breisgau, Germany

Abstract

This paper introduces a basic concept in VLSI layout which can find applications to submicron design, quantum devices, and designing new fine-grain digital, analog and mixed FPGAs. This concept is called Lattice Structure and it extends the concepts from [8] and [1, 13, 14, 16, 17]. In a regular arrangement of cells, every cell is connected to 4, 6 or 8 neighbors and to a number of vertical, horizontal and diagonal buses. Methods for expanding arbitrary binary, multivalued, and analog functions to this layout are illustrated.

Introduction.
Every signal in our layout, a Lattice, can be treated as continuous or multi-valued (particularly, binary). A multivalued connection for logic with 2^k values can be realized by k binary wires that go together (making the lattice "fat") and encode the multi-valued signal to binary. The well-known: Fat Trees, Generalized PLAs, Maitra cascades, and Akers Arrays [1, 4, 17] structures are only few special cases of this powerful concept. Our regular structure concept extends also some structures that exist in several patents of fine grain FPGAs (Motorola, Atmel [2], Plessey, Pilkington), but in addition we show constructive and efficient methods of designing discrete and continuous functions in these structures. We showed on many examples, [8, 10, 13, 14, 16, 17, 20] that this geometry is very powerful and better than the previously investigated general cellular structures. Here we will further extend and unify these notions to expansions with more than 2 successor functions, and geometries with more than 4 neighbors. In theory the lattices can be extended to any number of neighbors of a cell, but 8 is practically enough.

The fundament of our approach are **expansions** of functions, i.e. operators that transform a function to few simpler functions. There are two types of expansions: **canonical** (such as Shannon) and **noncanonical** (such as Sum-of-Products expansion). Also, the expansions can be characterized as of **maximum-type**, or of **Linearly-Independent type.** Maximum type expansions are generalizations of Shannon (S) [18], Post [10], and Sum-of-Products (SOP) [15] expansions. Linearly Independent expansions are generalizations of Davio expansions [18], and are for arbitrary algebras that have at least one linear operation, but most often are based on the algebraic structure of a **field** [6, 7, 8, 9, 10, 11, 13, 14].

These expansions are next mapped to neighborhood structures that are more powerful than those investigated theoretically in the past [1, 4], but similar to those from commercial Fine Grain FPGAs, [2].

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 nodes of a tree's 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. Below, we will present new geometries, expansions, and joinings operations on nodes. We will illustrate these concepts with just few simple examples of applications.

Layout Geometries.
In **case of 4 neighbors** (Fig. 1a,c,d,g) the lattice is planar and based on a rectangular grid. Each cell has two inputs (from N and E) and two outputs (to S and W), [1, 4, 17, 20]. Our structure generalizes the known switch realizations of symmetric binary functions [1, 4], based on Shannon expansion, but it allows also for Positive and Negative Davio expansions [18], negated variables and constants as control variables of the nodes, nodes controlled by not variables but functions, and inverted edges between nodes. Lattice diagram counterparts of Kronecker [7], Pseudo-Kronecker [18] and Free Diagrams can be realized for functions. It can be shown [8] that every function that is not symmetric can be symmetrized by repeating variables in the lattice layers, and the selection of the next variable is done using the Repeated Variable Maps from [12]. With respect to possible technological realization, the condition of only a single control variable in a level [4, 14] is no longer required, and all three types of buses (vertical, horizontal and diagonal) are used to lead any variable to the circuit's levels. Similarly we do not always require the existence of the diagonal buses, as well as the condition of having only constant values on the envelope of the circuit, or having the outputs only on the envelope. Now, arbitrary variables can occur on the envelope, and outputs can be taken from the middle by use of buses (this is an approach from commercial Fine Grain FPGAs [2]). Moreover, pairs or triplets of binary control variables can be used in nodes for arbitrary Linearly-Independent expansions [10]. Or equivalently, multivalued controls are used. In short, for a 4-neighbor lattice geometry, any canonical form of Reed-Muller logic and its Linearly Independent generalizations can

be realized (recall that the Reed-Muller logic is a special case of Galois Field logic, the GF(2)). Also, any MV logic (for instance in GF(4), [6]) can be also realized in the 4-neighbor lattice, but this would often require many repetitions of variables. Continuous and fuzzy functions can be also realized: in one example we realized an arbitrary piece-wise continuous function with a generalized Shannon expansion, [17].

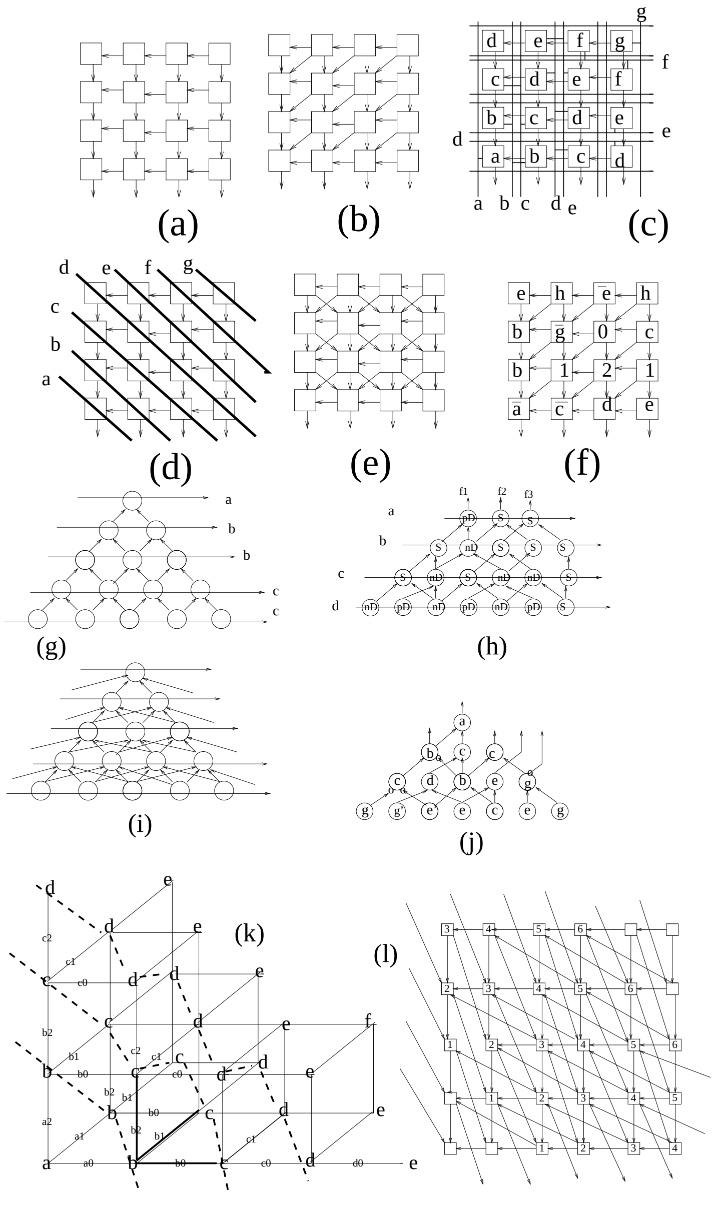

Figure 1: *Examples of regular lattices.*

In **case of 6 neighbors** (Fig. 1b,f,h,j,k) the rectangular grid is enhanced with one diagonal connection, thus every cell has three inputs (from N, NE and E) and three outputs (to W, SW and S). This allows to realize the generalized ternary diagrams (for binary EXOR logic) introduced in [9], as well as realizations of arbitrary expansion-based Post logic or GF(3) logic functions, [10] Finally, any binary, or MV logic can be mapped as in the case of the 4-neighbor lattice.

In **case of 8 neighbors** (Fig. 1e,i,l) the rectangular grid is enhanced with one more diagonal connection, so that every cell has four inputs (from N,NE,NW, and E), and four outputs (to S, SW, SE, and W). This allows to realize the generalized quaternary diagrams (for GF(4)), as well as realizations of arbitrary expansion-based Post or GF(k), $k < 4$ functions. Again, any binary or MV logic can be mapped, but more efficiently. In other variant, the neighborhoods are: NWW, NW, NE, NEE for inputs, SWW,SW SE, and SEE for outputs.

Especially practical among our examples are new ternary diagrams [9, 19] and Galois(4) expansions for multi-output incompletely specified functions. Our methods are very efficient for strongly unspecified functions. The introduced by us families of lattice diagrams are counterparts and generalizations of several diagrams known from the literature (BDDs, FDDs, KFDDs) as their subsets. Due to this property, our diagrams can provide a more compact representation of functions than either of the standard decision diagrams, because they do not require any routing or placement. Logic synthesis performs placement and routing as well. Lattice Kronecker Decision Diagrams (LKDDNEs) with negated edges are based on three orthogonal expansions (Shannon, Positive Davio, Negative Davio), and are created for incompletely specified Boolean functions as well. Many other new families of functional decision diagrams were created, including: Pseudo Kronecker Lattice DDs, Free Kronecker Lattice KDDs, Boolean Ternary Lattice DDs, and other [13]. The Boolean Ternary Lattice DDs introduce nodes with three edges and requires AND, OR and EXOR gates for expansion circuit realizations [14, 19]. Galois GF(3) and GF(4) diagrams require three and four neighbors on inputs, respectively.

Expansions and Joinings Operations on nodes.

There are two types of expansions: **maximum-type**, and **Linearly-Independent (LI) type.** Linearly Independent expansions are generalizations of Positive (pD) and Negative Davio (nD) expansions [18], and are for arbitrary algebras that have at least one linear operation, but most often are based on fields. They include S, pD, nD, Linearly Independent (binary, MV) [10], and EXOR Ternary [9, 19]. Joinings

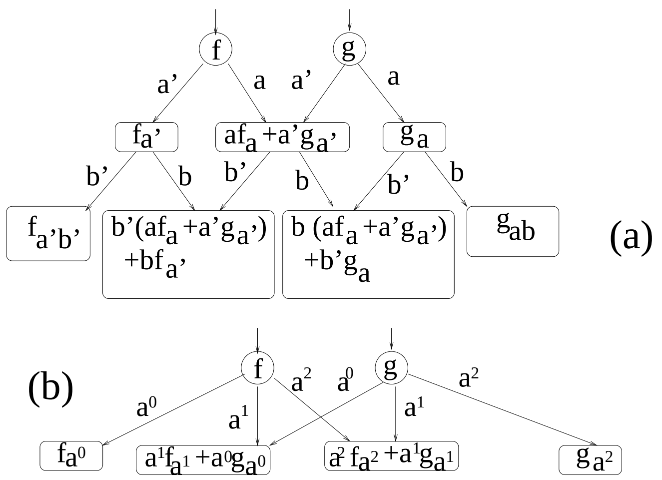

Figure 2: *Expansions and Joinings: (a) Shannon, (b) Ternary Post.*

operations for these expansions are more complicated and are presented in [8, 14].

Below we will briefly present only the maximum-type operations: Shannon, SOP, and Post (MV Shannon). We assume that each binary function f is represented by a pair [ON(f),OFF(f)]. Thus all cofactors f_a for product of literals a, are pairs: f_a = [ON(f_a),OFF(f_a)]. Fig. 2a explains the principle of creating a Shannon Lattice based on ordered Shannon expansions for two functions, f and g. Observe that every cofactor f_a of product a of an (in)complete function f can be interpreted as intersecting f with a and replacing all Kmap cells outside product a with don't cares. A standard cofactor f_x where x is a variable does not depend on this variable. In our interpretation, though, f_x is still a function of all variables including x, but as a result of cofactoring the variable x becomes vacuous. We will call this a **vacuous cofactor**, and denote by **v-cofactor**. Thus, for any two disjoint products a_1 and a_2, the v-cofactors f_{a_1} and g_{a_2} are disjoint (observe that standard cofactors are in general not disjoint), Therefore functions f_{a_1} and g_{a_2} are an incomplete tautology and functions f and g are not changed when f_{a_1} and g_{a_2} are joined (OR-ed) to create a new function. $a_1 f_{a_1} + \overline{a_2} g_{a_2}$, as in Fig. 2a (where $a_1 = a_2 = a$). This way, the entire lattice is created level-by-level, only two levels shown in Fig. 2a. Observe that functions inside the lattice become more and more unspecified when variables in levels are repeated, and ultimately they become constants, which terminates the expansion process.

In case when the products a_1 and a_2 are **not** disjoint, the v-cofactors f_{a_1} and g_{a_2} can be still in some cases an incomplete tautology. When they are a tautology, then functions f_{a_1} and g_{a_2} can be joined (OR-ed) without changing functions f and g. This way, because every variable cuts a Kmap into two disjoint parts, arbitrary two functions f and g can be always expanded together to a Shannon lattice, with OR-ing as a join operation. provided that the same variable x_i is used in the level, and all expansions use negated literal $\bar{x_i}$ in the left, and positive literal x_i of the variable in the right. As seen in Fig. 2, arbitrary functions of the same arguments are cut in half by expansion of each variable and new functions in levels are created by rearranging the cofactors in joinings. This process can lead to a slight increase of the number of nodes in comparison with a shared OBDD of these functions. But a regular structure is created, thus simplifying layout and making delays predictable.

The method to create Shannon Lattices can be easily expanded to MV Shannon expansions and associated Post Lattices for mult-output incomplete functions (see Fig. 2b). In ternary logic, each single-variable expansion cuts a function's map to three v-cofactors, and any two of them can be next recombined by a joining operation MAX - Fig. 2b. MAX is the maximum operation denoted by +.

In binary SOP expansions [15] a branching from node f is for any subset of literals l_j that their union covers the node function f. The SOP expansion is: $f = l_j f_{l_j} + l_r f_{l_r} .... + l_s f_{l_s}$. Of course, trees, diagrams and lattices based on SOP (binary and MV) expansions are not ordered, and are not canonical.

The method to create ordered Shannon lattices presented above can be expanded to free (non-ordered) Shannon Lattices and SOP Lattices. Any two nodes from the expansion that form an incomplete tautology can be joined as shown above.

The presented methods can be used for various kinds of ordered Shannon, and their generalizations can be used for arbitrary maximum-type and LI-type lattices. In case of fuzzy lattices, $x_i \cap \bar{x_i} \neq 0$ so the cofactors are not disjoint, but the method can be still used provided that special fuzzy literals $fflx_i$ are assumed that are disjoint.

Design methodology.
One level of multi-output function is expanded to an assumed type of a Decision Tree (Shannon, SOP, Post, etc). The selection of a good order of variables and expansion nodes is based on generalized partial symmetries for cofactors (there are for instance as many as 81 generalized symmetries for Galois Field(2)) and thousands of symmetries for GF(3)) [5]. In the next phase, the level of the tree is mapped to the assumed type of Lattice. This means joining together some nodes of the tree-like lower part of the lattice. The procedure requires repeating some variables in the lattice. We show that for real-life benchmark functions, and starting from the decompositional hierarchy of partitioning variables [12], the overhead of variable repeating is not excessive in each decomposed block [5, 14]. This is because symmetric and nearly symmetric blocks are preferred by our decomposer [12].

Calculation of data input functions to lattice nodes is done using a technique very similar to the one based on Linearly Independent logic from [11]). Selection of the order of (repeated) variables [5] is done using the concept of best minterm separation using the Repeated Variable Maps from [12].

Examples of structures and expansions.
Few figures briefly illustrate various applications of our ap-

proach. Figure 1a presents the standard lattice structure investigated in [1, 4, 17]. Figure 1b presents this structure expanded with NE/SW diagonal connections, and Figure 1e presents this structure expanded with NE/SW and NW/SE diagonal connections. Figure 1c illustrates that standard vertical and horizontal buses can be used instead of diagonal buses for small functions. Figure 1d shows a symmetric function of 7 variables realized in a standard lattice structure with diagonal buses (no repeated variables). Figure 1f shows a non-symmetric function realized in a standard lattice structure with control input run from diagonal, vertical and horizontal buses (buses not shown, please observe the constants 0 and 1 inside the lattice, as well as the folded and repeated variables).

Figure 1g shows another view of a standard lattice for symmetricized non-symmetric function (variables b and c are repeated). Figure 1h presents a lattice with 3x3 neighborhood and repeated and folded variables. Figure 1i presents a lattice with 4x4 neighborhood. Another 4x4 lattice is shown in Figure 1d. Figure 1j presents a lattice with 3x3 neighborhood drawn to emphasize ternary symmetries in 3-dimensional space. For each node, its expansion variable is shown. Some edges show variables' values. Bold edges in the figure correspond to three values of variable b expanded for $a = 0$. Observe, that the same figure can be also looked at as a flat connection plan and diagonal NW/SE buses can be added (interrupted lines). Observe also, that in this figure the number of nodes in the layers is 1,3,6,9,... while in the regular lattice from Figure 1h the numbers were 1,3,5,7... so that not all partial ternary symmetries could be realized there in one level. This is an example of a trade-off between the geometry and logic, that must be solved in selecting the type of lattice for a type of function.

Figure 3 presents a comparison of sizes of a standard Shannon lattice and our lattices. Figure 3a presents a solution that would be obtained using the method from [4]. The shape is a trapezoid and the size is 14 nodes. Connectivity pattern is 2x2. The Akers Array would have $(5 * 5) * 2$ nodes (it realizes each of two functions separately, and uses a $5 * 5$ fixed rectangle for a 4 variable function). Figure 3b presents our solution with 4x4 connectivity pattern array of multiplexers. It is linear in shape and has $2 * 4 = 8$ nodes. In addition to Shannon, the Shannon expansions with negated control variables are now used. Figure 3c presents our solution with 2x2 connectivity pattern array of positive Davio nodes. It is nearly linear in shape and has 5 nodes. Predictability and equality of delays should be appreciated in all lattices.

Figure 4 presents different expansion nodes for various kinds of expansions for binary, multi-valued, fuzzy and continuous analog functions: (a) two views of a cell, a multiplexer, and a general 2x2 cell in a Lattice that may be realized by this mux (the notation of inputs and outputs is preserved in next examples); (b) positive Davio expansion node, (c) negative Davio expansion node, (d) fuzzy logic expansion node, (e) Shannon node for ternary logic, (f) Shannon node for quaternary logic, (g) realization of the Shannon node from (f) with binary logic, (h) expansion node for fuzzy logic with arbitrary literal functions.

Iterative Circuits, Circuits with Memory, and Analog Design.

Hierarchical design is possible of **iterative** one- and two-dimensional structures, which are cellular connections of blocks, each block realized as a lattice. This can be done also for discrete circuits with memory. Analog counterparts use sample-hold analog memories, which play the same role as flip-flops in discrete technologies. The introduced Lattice Structures allow therefore to realize cellular memory-less functions, finite state machines, and infinite state machines, realized in analog, binary, or multivalued logic. For instance, the digital and analog: filters, pipelined image processors, or systolic processors can be the examples. We showed how an elliptic filter in OTA technology can be mapped to this structure [16]. Fuzzy memory-less arrays are presented in [17]. Fig. 4a shows a basic circuit with analog comparator and analog multiplexer, and Fig. 4b a full cell with SRAM-controlled muxes to switch the inputs. Two simple lattices for analog functions are shown in Figure 5c,d. Fig. 4c presents a lattice realization of the piecewise continuous function **if** $((c > d)$ **and** $(a > b))$ **then y else if** $((a \leq b$ **and** $(c \leq d))$ **then cos(y) else sin(y)**. Fig. 4d shows a lattice realization of analog function $\max(h_1, h_2, h_3, h_4, h_5)$. Similar realizations can be created for rank and median filters, cellular neural nets, equation solvers, and (analog and digital) image processing circuits.

Conclusion.

In conclusion, the main idea of the presented approach can be summarized as follows: starting from **all possible** neighbor geometries in two and three dimensional spaces, we create **all possible regular structures**. This is more powerful than in the previous structures [1, 4] which considered limited planar geometries. Next we design **arbitrary expansions** for any of the structures. New expansions can be constructed based on the Linearly-Independent function theory, or any other canonical or non-canonical function expansions. There is a very high number of various new expansions, in contrast to only Shannon expansion types used in lattice diagrams from [1, 4, 5]. This way, the same, in principle, layout-driven synthesis approaches are created for binary, multivalued, linearly-independent, Galois, fuzzy and analog functions,[8, 17]. Thus, the presented approach generalizes and unifies many known expansions, decision diagrams, and regular layout geometries. These methods are of special interest to deep sub-micron technology and binary and MV pass-transistor design as well as OTA design.

References

[1] S.B. Akers, "A rectangular logic array," *Trans. on Comp.*, Vol. C-21, pp. 848-857, August 1972.

[2] Concurrent Logic Inc., "CLI 6000 Series Field Programmable Gate Arrays," *Prelimin. Inform.*, Dec. 1, 1991, Rev. 1.3.

[3] R.E. Bryant, "Graph-based algorithms for boolean function manipulation, *IEEE Trans. Comp.*, C-35, No. 8, pp. 667-691, 1986.

[4] M. Chrzanowska-Jeske, Z. Wang and Y. Xu, "A Regular Representation for Mapping to Fine-Grain, Locally-Connected FPGAs," *Proc. ISCAS'97.*

[5] M. Chrzanowska-Jeske, M.A. Perkowski, and Y. Xu, "Lattice Diagrams for Deep Sub-Micron Technology," *submitted to ICCAD'97.*

[6] K.M. Dill, K. Ganguly, R. Safranek, and M.A. Perkowski, "A New Linearly Independent, Galois-Field Reed-Muller Logic," *submitted to Reed-Muller'97.*

[7] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski, "Efficient representation and manipulation of switching functions based on Ordered Kronecker Functional Decision Diagrams," *Proc. DAC*, pp. 415-419, 1994.

[8] M.A. Perkowski, and E. Pierzchala, "New Canonical Forms for Four-valued Logic", *Report, Dept. Electr. Engn.* PSU, 1993.

[9] M. Perkowski, M. Chrzanowska-Jeske, A. Sarabi, and I. Schaefer, "Multi-Level Logic Synthesis Based on Kronecker and Boolean Ternary Decision Diagrams for Incompletely Specified Functions," *VLSI Design*, Vol. 3, Nos. 3-4, pp. 301-313, 1995.

[10] M. Perkowski, A.Sarabi, and F. Beyl, "Fundamental Theorems and Families of Forms for Binary and Multiple-Valued Linearly Independent Logic," *Proc. Reed-Muller'95*, pp. 288-299.

[11] M. Perkowski, L. Jozwiak, and R. Drechsler, "A Canonical AND/EXOR Form that includes both the Generalized Reed-Muller Forms and Kronecker Reed-Muller Forms," *submitted to Reed-Muller'97.*

[12] M. Perkowski, M. Marek-Sadowska, L. Jozwiak, T. Luba, S. Grygiel, M. Nowicka, R. Malvi, Z. Wang, and J. Zhang, "Decomposition of Multi-Valued Relations," *Proc. ISMVL'97*, Nova Scotia, May 1997, pp. 13 - 18.

[13] M.A. Perkowski, L. Jozwiak, and R. Drechsler, "Two hierarchies of Generalized Kronecker Trees, Forms, Decision Diagrams, and Regular Layouts," *submitted to Reed-Muller'97.*

[14] M.A. Perkowski, M. Chrzanowska-Jeske, and Y. Xu, "Binary Lattice Diagrams for Reed-Muller Logic Realization in Regular Layouts with Predictable Timing," *submitted to Reed-Muller'97.*

[15] M.A. Perkowski, "Bidirectional Decision Diagrams for Synthesis with Complex Pass Transistor Gates," *Booklet Intern. ULSI Workshop*, Antigonish, Nova Scotia, May 27, 1997, pp. 37 - 40.

[16] E. Pierzchala, and M.A. Perkowski, "High Speed Field Programmable Analog Array Architecture Design," *Proc. FPGA'94*, Berkeley, California, February 1994.

[17] E. Pierzchala, M.A. Perkowski, and S. Grygiel, "A Field Programmable Analog Array for Continuous, Fuzzy, and Multi-Valued Logic Applications," *Proc. 24-th ISMVL*, pp. 148-155, Boston, May 25-27, 1994.

[18] T. Sasao (editor), "Logic Synthesis and Optimisation," *Kluwer Academic Publishers,* 1993.

[19] T. Sasao, "Ternary Decision Diagrams: Survey," *Proc. ISMVL'97*, pp. 241 - 250.

[20] A. Sarabi, N. Song, M. Chrzanowska-Jeske, and M. A. Perkowski, "A Comprehensive Approach to Logic Synthesis and Physical Design for Two-Dimensional Logic Arrays," *Proc. DAC '94,* San Diego, June 1994, pp. 321 - 326.

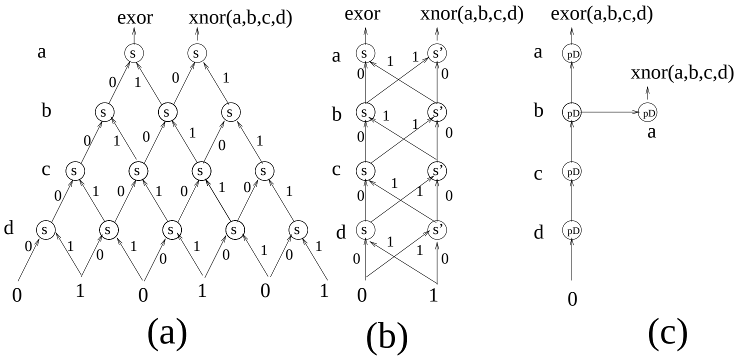

Figure 3: *Comparison of three types of Lattices for two-output EXOR/XNOR function.*

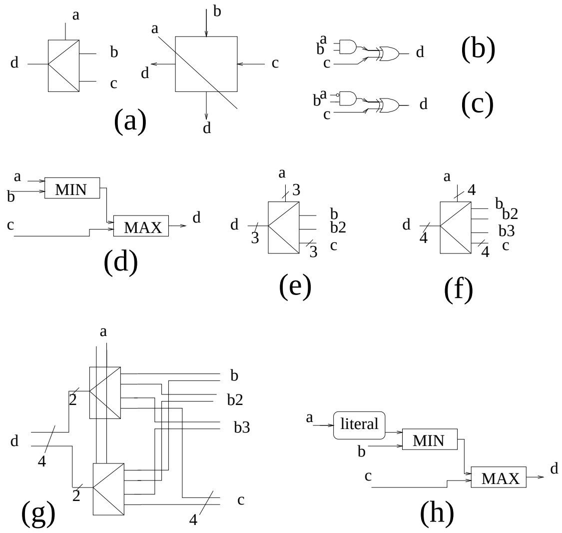

Figure 4: *Comparison of Expansion nodes for lattices.*

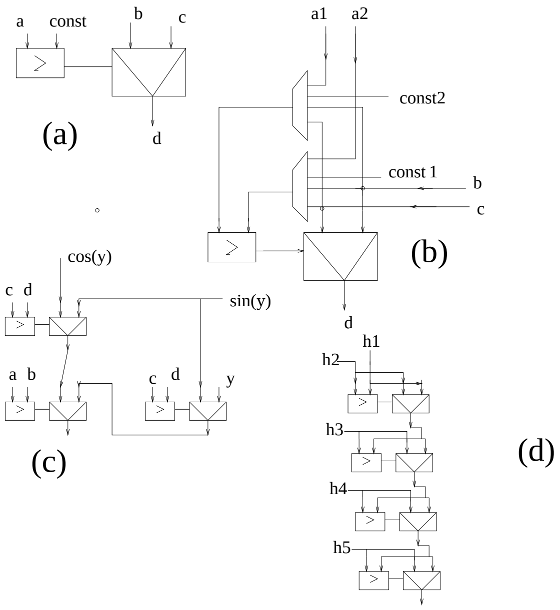

Figure 5: *Realization of analog functions.*