NEW HIERARCHIES OF AND/EXOR TREES, DECISION DIAGRAMS, LATTICE DIAGRAMS, CANONICAL FORMS, AND REGULAR LAYOUTS

Marek Perkowski, Lech Jozwiak †, Rolf Drechsler +,

Portland St. Univ. Dept. Electr. Engn., Portland, USA
† Faculty of Electr. Engng., Eindhoven Univ. of Techn., The Netherlands,
+ Inst. of Comp. Sci., Albert-Ludwigs Univ., Freiburg in Breisgau, Germany,

MAIN CONTRIBUTIONS OF THIS PAPER

- Generalizations of the **Generalized Kronecker** representations.
- New canonical AND/EXOR **forms**.
- **Interrelated hierarchies** of canonical AND/EXOR trees, decision diagrams, lattice diagrams, canonical forms and regular layouts.
- The new diagrams and forms can be used for synthesis of quasi-minimum **Exclusive Sum of Products** (ESOP) circuits,
- Highly **testable** multi-level AND/EXOR circuits, that through the *"EXOR-related technology mapping"* are adjusted to AND/OR/EXOR custom VLSI, standard cell, or other technologies.
- Applications in Fine Grain Field Programmable **Gate Arrays**.

MAIN CONTRIBUTIONS OF THIS PAPER (cont)

- The new diagrams can represent **large** functions.
- **Lattice Hierarchy** generalizes and extends the Universal Akers Array to expansions other than Shannon and neighborhoods other then 2-inputs, 2-outputs.
- These Lattice Diagrams find many applications in **layout-driven logic synthesis**, particularly for XILINX FPGAs.

ACKNOWLEDGE CONTRIBUTIONS OF ZHEGALKIN

- Zhegalkin in 1927 discovered the forms, now attributed to Reed and Muller and invented by them in 1954.
- His contributions are not properly acknowledged.
- As a community, it is fair to acknowledge him, as we had acknowledged Davio, Reed and Muller.
- We propose to call all these new forms that properly include both KRO and GRM, as well all the future AND/EXOR forms including these two families concurrently, **the Zhegalkin forms.**

PLAN

- Various kinds of trees with multi-variable nodes.
- Review the concept of the Generalized Kronecker Trees.
- Generalized Kronecker Diagrams and their "pseudo"-like generalizations.
- Generalized Kronecker Forms.
- Extended Green/Sasao hierarchies of trees, forms and Diagrams.
- New Hierarchy of Zhegalkin Lattice Diagrams.
- Current and Future work.
- Conclusions.

KRONECKER FAMILIES OF REPRESENTATIONS

- Decision Diagrams (DDs).
- Used in logic synthesis, verification and simulation.
- Main internal representation of functions, on which all meaningful operations are executed.
- DDs originate from binary decision trees (binary expansion trees, Shannon trees), which in turn are based on the **fundamental expansion theorem of Shannon** that is applied in every node of a tree.
- Every node is related to one input variable of the function.
- **"Reduced"**, and **"Ordered"**.
- Disadvantage - large functions cannot be represented.

- Generalise the concept of a binary tree!!
- **Green/Sasao hierarchy** of representations characterizes these representations in an **uniform way** that we will **use here** in our generalizations.
- All these representations can be used in the first stage of logic synthesis - the "*technology independent, EXOR synthesis*" phase, which is next followed by the "*EXOR-related technology mapping*".

DAVIO EXPANSIONS

- The hierarchy is based on three expansions:

$$f(x_1, x_2, ..., x_n) = \overline{x}_1 f_0(x_2, ..., x_n) \oplus x_1 f_1(x_2, ..., x_n)$$

in short $f = \overline{x}_1 f_0 \oplus x_1 f_1$, called Shannon, (1.1)

$$f(x_1, x_2, ..., x_n) = 1 \cdot f_0(x_2, ..., x_n) \oplus x_1 f_2(x_2, ..., x_n)$$

in short $f = f_0 \oplus x_1 f_2$, called Positive Davio, (1.2)

$$f(x_1, x_2, ..., x_n) = 1 \cdot f_1(x_2, ..., x_n) \oplus \overline{x}_1 f_2(x_2, ..., x_n)$$

in short $f = f_1 \oplus \overline{x}_1 f_2$, called Negative Davio, (1.3)

where f_0 is f with x_1 replaced by 0 (negative cofactor of variable x_1), f_1 is f with x_1 replaced by 1 (positive cofactor of variable x_1), and $f_2 = f_0 \oplus f_1$.

ORDERED EXOR-BASED REPRESENTATIONS

- By applying recursively expansions (1.1) - (1.3) (or any subset of them) to the function various types of binary decision trees can be created.
- The concepts of:
  - **Shannon Trees**, **Positive Davio Trees,**
  - **Negative Davio Trees,**
  - **Kronecker Trees**,
  - **Reed-Muller Trees,**
  - **Pseudo-Kronecker Trees,**
  - **Pseudo Reed-Muller Trees**,
  - as well as of the corresponding **decision diagrams**,
  - and flattened (two-level) **canonical forms.**

FREE EXOR-BASED REPRESENTATIONS

- **Free Kronecker Trees** use S, pD and nD nodes **disregarding any order** of variables and expansions (Ho/Perkowski).
- At every tree level, different variables and expansions can occur.
- Thus, the order of variables in every branch can be different, and such diagrams are also called non-ordered.
- Similarly, one can also define Free Binary Decision Trees (leading to Free BDDs) and Free Positive Davio Trees (leading to Free FDDs).
- Free Kronecker Trees lead to **Free KFDDs** (FKFDDs) (Ho/Perkowski).

OUR NEW GENERALIZATIONS

- Use **binary operators**, so useful for logic synthesis.
- **Generalized Kronecker** Trees, Forms and Decision unify Kronecker and Generalized Reed Muller representations (Perkowski/Jozwiak/Drechsler).
- Here we **further generalize** the Generalized Kronecker Trees, Forms and Decision Diagrams.
- These representations are **better then all Kronecker-like representations** because they did not allow to create GRM forms after flattening.
- We create an **enhanced** Green/Sasao hierarchy.
- Because of the superioriority of new representations, the circuits are also **never worse** than the AND/EXOR circuits obtained from the **previous** representations, (including the GKTs).

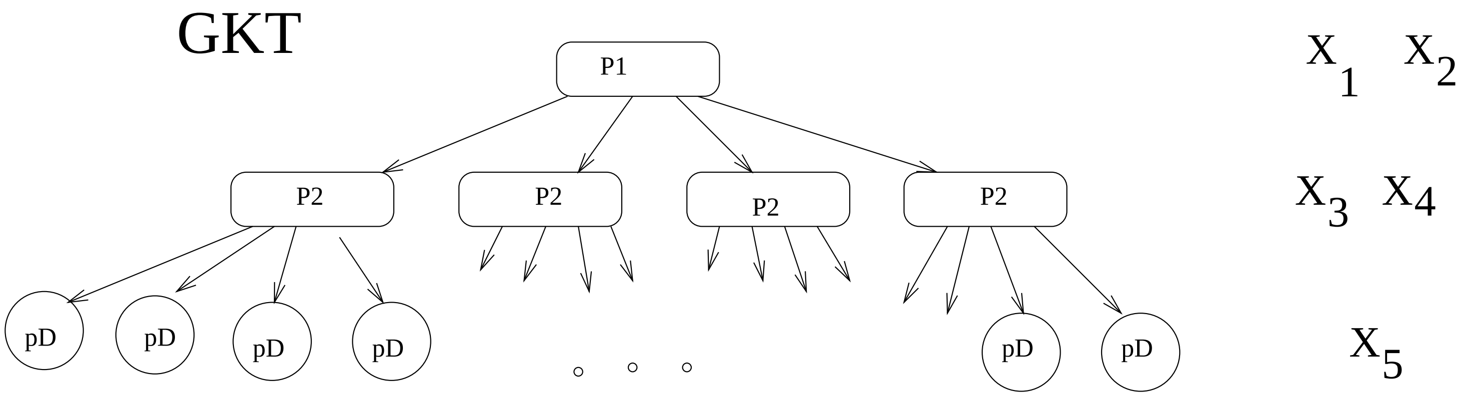

Figure 1: *Generalized Kronecker Tree*

GENERALIZED REED MULLER EXPANSION WITH CONSTANT COEFFICIENTS

- $f(x_1, \ldots, x_n) = a_0 \oplus a_1\hat{x}_1 \oplus a_2\hat{x}_2 \oplus \ldots \oplus a_n\hat{x}_n \oplus a_{12}\hat{x}_1\hat{x}_2 \oplus a_{13}\hat{x}_1\hat{x}_3 \oplus \ldots \oplus a_{n-1,n}\hat{x}_{n-1}\hat{x}_n \oplus \ldots \oplus a_{12\ldots n}\hat{x}_1\hat{x}_2\hat{x}_3 \ldots \hat{x}_n \quad (2.1)$ where a_i's are either 0 or 1, and $\hat{x}$ denotes variable x or its negation, $\overline{x}$.

- By assigning a variable or a negation of a variable to each of the $\hat{x}_i$ in (2.1) we create $2^{n2^{n-1}}$ different expansion formulas.
- Each of them is called **a polarity expansion**, i.e., an expansion of a certain polarity.

GRM EXPANSION WITH FUNCTIONAL COEFFICIENTS

- In every node of GKT, the formula for expansion (2.2) is applied which generalizes the formula (2.1) for a Generalized Reed-Muller expansion.
- The expansion formula (2.2) for function $f(x_1, x_2, ..., x_m, ..., x_n)$ is the same as GRM expansion, (2.1), with respect to variables $x_1, ..., x_m$ and coefficients a_i replaced with subfunctions SF_i of remaining variables $x_{m+1}, ... x_n$:

$$f(x_1, x_2, ..., x_m, ..., x_n) = SF_0(x_{m+1}, ..., x_n) \oplus$$
$$\hat{x}_1 SF_1(x_{m+1}, ... x_n) \oplus \hat{x}_2 SF_2(x_{m+1}, ..., x_n) \oplus \ldots \oplus \hat{x}_m SF_m(x_{m+1}, ..., x_n)$$
$$\oplus \hat{x}_1 \hat{x}_2 SF_{12}(x_{m+1}, ..., x_n) \oplus \hat{x}_1 \hat{x}_3 SF_{13}(x_{m+1}, ..., x_n)$$
$$\oplus \ldots \hat{x}_{m-1} \hat{x}_m SF_{m-1,m}(x_{m+1}, ..., x_n)$$
$$\oplus \ldots \oplus \hat{x}_1 \hat{x}_2 \hat{x}_3 \ldots \hat{x}_m SF_{12 \ldots n}(x_{m+1}, ..., x_n) \qquad (2.2)$$

GRM EXPANSION (cont)

where the so-called *Data Functions* SF_i are calculated as coordinates of a vector CV $= M^{-1} \times$ FV, in which:

$FV(x_{m+1}, ..., x_n)$ is a vector of cofactors of F with respect to variables from the set $\{x_1, ..., x_m\}$.

- A $2^m \times 2^m$ **nonsingular matrix** M has as its columns all the products of literals $\hat{x}_i$ that have been used in one of $2^{m2^{m-1}}$ particular polarity expansion forms specified by (2.2).
- The columns of the matrix are **linearly independent** with respect to the bit-by-bit exoring operation.

Definition 1. The **Generalised Kronecker Tree** is a multi-branch tree created as follows:

1) The set of all n input variables is **partitioned** into disjoint and nonempty subsets S_j such that the union of all these subsets forms the initial set. (If each subset includes just a single variable, the tree reduces to the special case of a KRO tree. If there is only one subset that includes all variables, the tree corresponds to the special case of a GRM.)
2) The sets are ordered, each of them corresponds to a level of the tree.
3) **When the set has one variable:** S, nD, or pD expansion is selected for all its nodes.
4) **When the set is multi-variable:** the formula for expansion (2.2) is applied with the same polarity for all variables from S_j.

PSEUDO GENERALIZED KRONECKER TREE

Definition 2. The **Pseudo Generalised Kronecker Tree** is a tree with multi-variable expansion nodes created as follows

1) The set of all n input variables is **partitioned** to disjoint and nonempty subsets S_j such that the union of these subsets forms the initial set (If each subset has just a single variable, the special case of PKRM is considered.)

2) For every node of the tree, in a level with variable subset S_j, **any** GRM expansion for all variables from S_j can be performed.

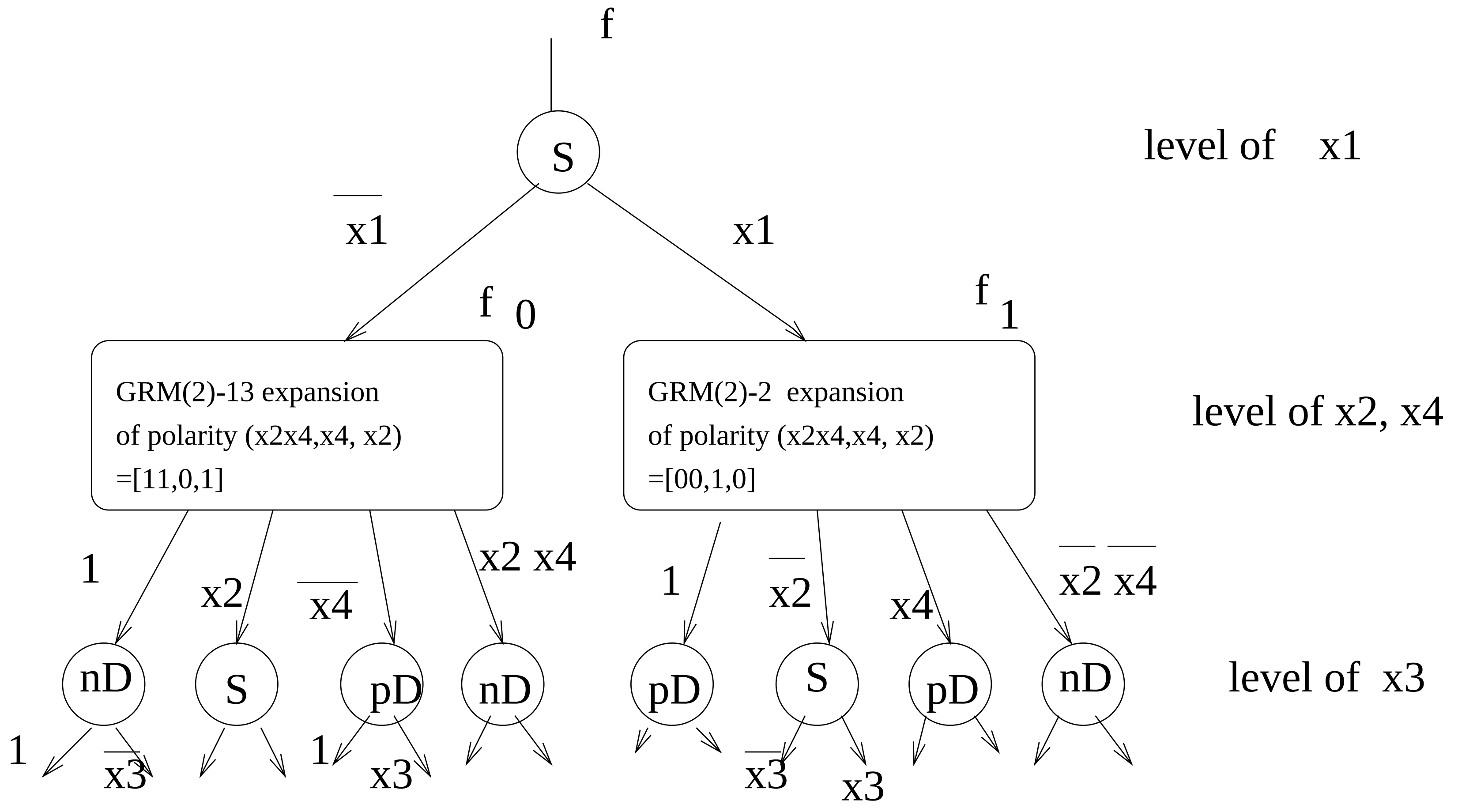

Figure 2: *Example of Pseudo Generalized Kronecker Tree*

NOTATION

- Number 13 in expansion name (polarity) "GRM(2)-13" is a natural number corresponding to the binary number 1101, called a **polarity**.
- The expansion of the node GRM(2)-13 is described by the following formula:

  $f_0(x_2, x_3, x_4) = SF(f_0)_1(x_3) \oplus x_2 SF(f_0)_{x_2}(x_3)$
  $\oplus \overline{x}_4 SF(f_0)_{\overline{x}_4}(x_3) \oplus x_2 x_4 SF(f_0)_{x_2 x_4}(x_3)$

  where notation $SF(f)_i(X)$ denotes function SF_i, with arguments from the set X of variables, applied to argument function f.
- The GRM expansion in node GRM(2)-2 (for polarity [00,1,0]) is described by the formula:

  $f_1(x_2, x_3, x_4) = SF(f_1)_1(x_3) \oplus \overline{x}_2 SF(f_1)_{\overline{x}_2}(x_3) \oplus x_4 SF(f_1)_{x_4}(x_3) \oplus \overline{x}_2 \overline{x}_4 SF(f_0)_{\overline{x}_2 \overline{x}_4}(x_3)$

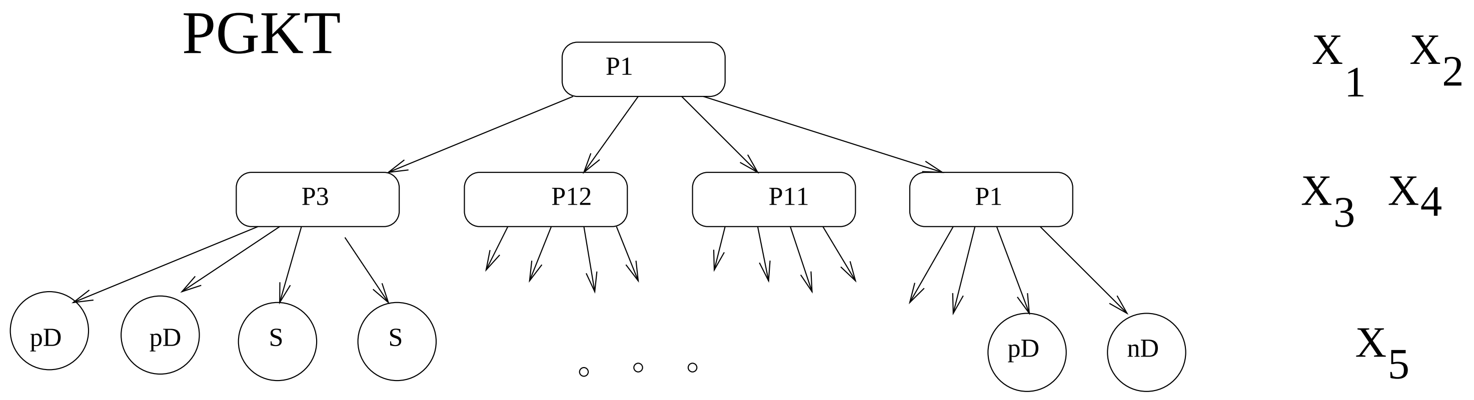

Figure 3: *Pseudo Generalized Kronecker Tree*

MIXED PSEUDO GENERALISED KRONECKER TREE

Definition 3. The **Mixed Pseudo Generalised Kronecker Tree** (MPGKT) is any multi-branch tree created as follows.

1) The set of all n input variables is **partitioned** to disjoint subsets (blocks) S_j of variables.

2) For every multi-variable set, either apply to a root node in the level **an** arbitrary GRM expansion of **all** its variables or create a subtree of single-variable expansions of variables from S_j.

Ordered sets S_j are assumed. For a single-variable level of the tree, any combination of the S, nD and pD nodes can be applied.

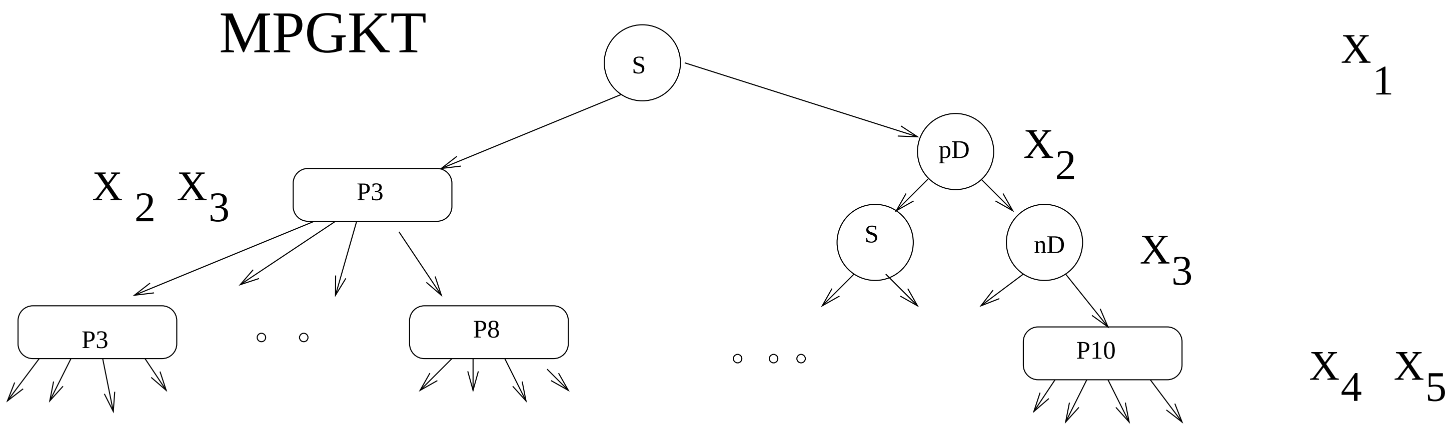

Figure 4: *Mixed Pseudo Generalized Kronecker Tree*

ORDERED GENERALISED KRONECKER TREE

- *Definition 5.* The **Ordered Generalised Kronecker Tree** (OGKT) is any multi-branch tree created as a Free Tree, with the additional constraint that every branch has the same order of variables.
- In OGKT the sets of variables in different branches may have different sizes and overlap, but the order must be the same.
- For instance, in one branch the first set is $\{x_0, x_1, x_2\}$ the second set is $\{x_3, x_4\}$, and the third set is $\{x_5, x_6\}$. In another branch the sets are: $\{x_0\}$, $\{x_1\}$, $\{x_2, x_3\}$, and $\{x_4, x_5, x_6\}$.

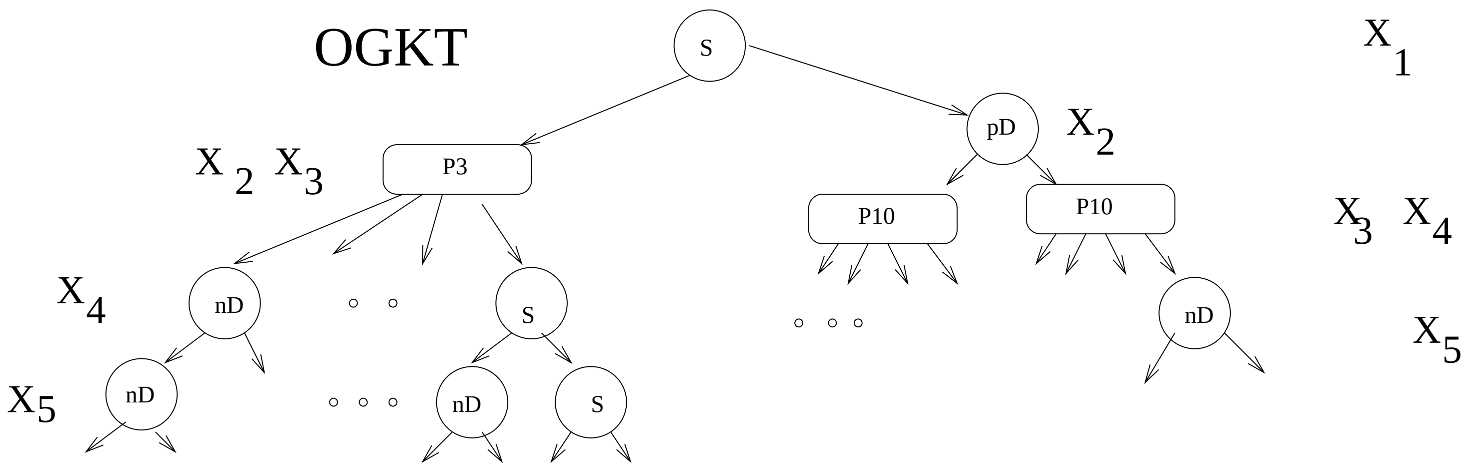

Figure 5: *Ordered Generalized Kronecker Tree*

FREE GENERALIZED KRONECKER TREES

Definition 4. The **Free Generalised Kronecker Tree** (FGKT) is any multi-branch tree created as follows:

- For every node of the tree, **an** arbitrary size subset of input variables can be selected.
- For single variable sets - an S, pD or nD node can be created, for multi-variable sets - the GRM expansion of its variables of any polarity is calculated.
- The levels are no more associated with variables or their sets, various local orders and partitions of variables may exist in the branches.

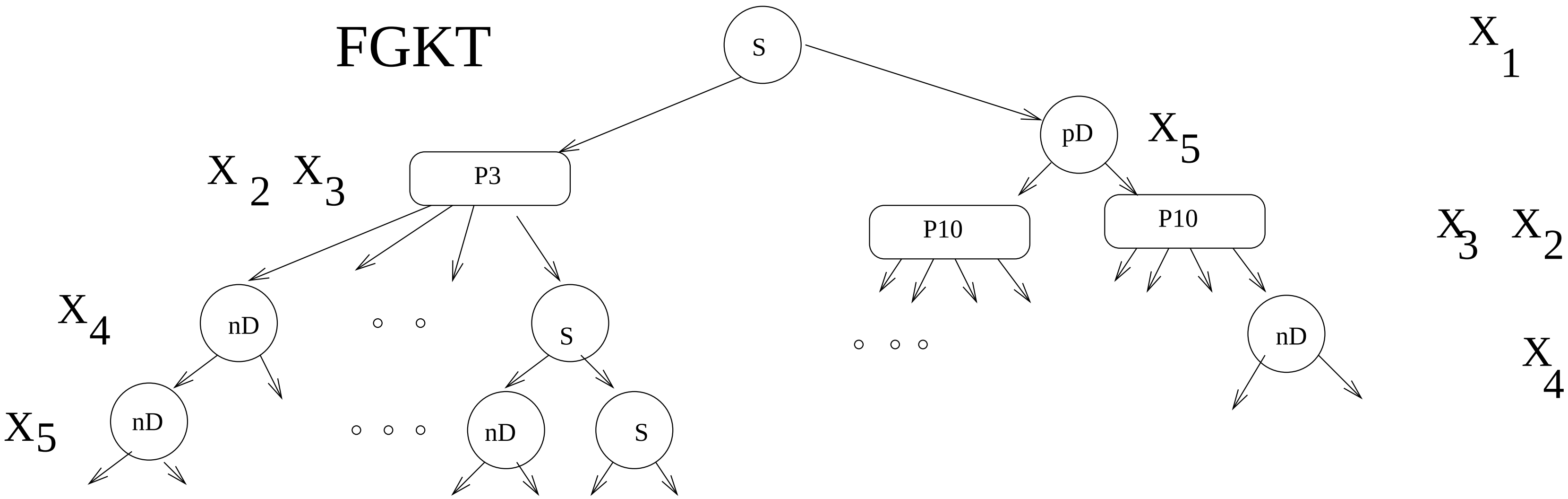

Figure 6: *Free Generalized Kronecker Tree*

ZHEGALKIN FORMS

- **Zhegalkin forms** are **all** AND/EXOR forms that **properly** include **both** the KRO and the GRM forms.
- All Zhegalkin forms are defined here for the first time.
- The simplest Zhegalkin form is the Generalized Kronecker form.
- Of course, every particular GRM or KRO form, being a special case of a Generalized Kronecker form, is in this sense also a Zhegalkin form.
- We propose, however, that when we will be talking about existing **families of forms**, we will keep the known names without the name "Zhegalkin"
  (Similarly, every FPRM form is a Kronecker form, but the Fixed-Polarity Reed-Muller family of forms is distinguished from the broader Kronecker family of forms).
- The forms presented here are **straightforward** generalization of the known families.

DEFINITION OF ZHEGALKIN FORMS

- *Definition 7.* The canonical AND/EXOR Form corresponding to each of the above defined types of trees, **the Zhegalkin Form**, is obtained by flattening the respective tree.
- **Flattening** means finding the AND-terms by following all the paths from the root to all the leafs. This way, an AND/EXOR two level expression is created, that is equivalent to the tree and that is a canonical form.

REDUCED ZHEGALKIN FORMS

- GKT families in which for the **"Kronecker-type" variables** not all expansion types S, pD and nD are executed, but **only a subset** of them.
- *Definition.* The **pD/nD Fixed/Mixed Reed-Muller Expressions (forms)** (FMRMEs) are canonical forms obtained analogously to GKE, but instead of all Shannon, positive and negative Davio expansions, only positive and negative Davio expansions are used for the variables from the "Kronecker-type" set of variables.
- Because of the existence of efficient FPRM minimization algorithms, calculation of FMRME forms can be made more efficiently.
- Expansions for any non-empty subset of {S,pD,nD} expansions for "Kronecker-type" variables can be defined; {S,pD,}, {S,nD}, {S}, {pD}, and {nD}.
- Define **FMRM Trees** and **FMRM DDs**, analogously to the general cases of GKTs and GK DDs.

GENERALIZED KRONECKER DECISION DIAGRAMS

- *Definition 8.* **Canonical AND/EXOR Decision Diagrams** corresponding to each of the above defined types of trees are obtained by combining isomorphic nodes (i.e, the nodes that correspond to logically equivalent, tautological, subfunctions), and removing nodes that are redundant in the same sense as the one discussed for GKTs.
- For instance, a **node** with two inputs originating from the same node is **removed** for binary nodes.
- Similarly, a node with four inputs originating from the same node is removed for quaternary nodes.

GENERALIZED KRONECKER DECISION DIAGRAMS (cont)

- **Transformations** for S, pD and nD nodes (Drechsler) are applied to single-variable nodes.
- The transformations for **multi-variable** nodes are their straightforward generalizations.
- All these transformations generalize the OKFDD transformations.
- This way, the precise definitions of all GK decision diagrams and respective flattened (Zhegalkin) forms can be obtained.

APPLICATIONS OF NEW REPRESENTATIONS

(1) **Synthesis of easily testable**, two-level and multilevel, AND/EXOR circuits.

(2) **Representation of large functions.**

(3) **Synthesis of circuits with predictable and controllable timing.**

(4) **Synthesis for Fine Grain FPGAs** (Atmel) and **standard FPGAs** (XILINX).

(5) **Exact and approximate ESOP minimization.**

Type of Tree	Expression generated from the tree	Decision Diagram Generated from Tree
Shannon Tree	Minterm Expansion	Binary Decision Diagram (BDD)
Positive Davio Tree	Positive Polarity Reed-Muller (PPRM)	Functional Decision Diagram (FDD) Kebschull et al
Reed-Muller Tree	Fixed Polarity Reed-Muller (FPRM)	Reed-Muller Decision Diagram
Kronecker Tree	Kronecker Expansion (KRO)	Ordered Kronecker Functional Decision Diagram (OKFDD) Perkowski et al, Drechsler et al
Pseudo Reed-Muller Tree	Pseudo Reed-Muller Expansion (PSDRM)	Pseudo Reed-Muller Decision Diagram
Pseudo Kronecker Tree	Pseudo Kronecker Expansion (PSDKRO)	Pseudo Kronecker Decision Diagram (PKDD) Sasao
Free Kronecker Tree	Free Kronecker Expansion (FKE)	Free Kronecker Decision Diagram (FKFDD) Perkowski/Ho

Table 1: *Relations of Known Canonical AND/EXOR Trees, Expressions and Decision Diagrams.*

Type of Tree	Expression generated from the tree	Decision Diagram Generated from Tree
pD/nD Fixed/Mixed RM Tree	*pD/nD Fixed/Mixed Expansion*	*pD/nD Fixed/Mixed Lattice Diagram*
Generalized Kronecker Tree (GKT)	**Generalized Kronecker Expansion (GKE)**	Generalized Kronecker Decision Diagram (GKDD)
Pseudo Generalized Kronecker Tree (PGKT)	**Pseudo Generalized Kronecker Expansion (PGKE)**	Pseudo Generalized Kronecker Decision Diagram (PGKDD)
Mixed Pseudo Generalized Kronecker Tree (MPGKT)	**Mixed Pseudo Generalized Kronecker Expansion (MPGKE)**	Mixed Pseudo Generalized Kronecker Decision Diagram (MPGKDD)
Ordered Generalized Kronecker Tree (OGKT)	**Ordered Generalized Kronecker Expansion (OGKE)**	Ordered Generalized Kronecker Decision Diagram (OGKDD)
Free Generalized Kronecker Tree (FGKT)	**Free Generalized Kronecker Expansion (FGKE)**	Free Generalized Kronecker Decision Diagram (FGKDD)

Table 2: *Relations of New Canonical AND/EXOR Trees, Expressions and Decision Diagrams. An example of Fixed/Mixed representations is in italic, Zhegalkin forms are in bold.*

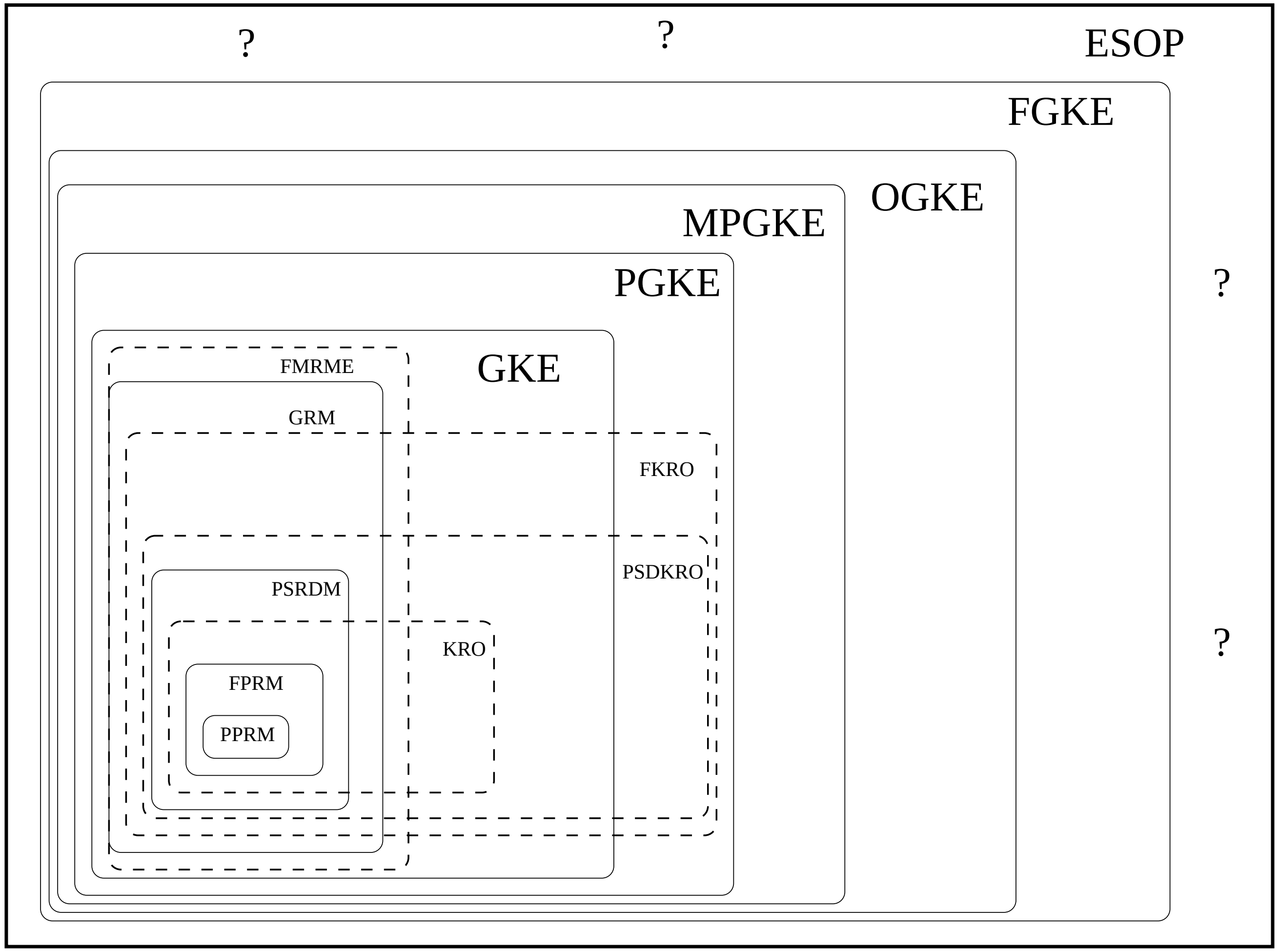

Figure 7: *Set-theoretical relationship among known and new (Zhegalkin) classes of AND/EXOR canonical forms*

Type of Tree	Lattice Diagram Generated from Tree and type of regular layout	Where lattices are discussed in more detail
Shannon Tree	Universal Akers Array Pseudo-Symmetric BDD (PSBDD)	Akers Chrzanowska
Positive Davio Tree	Functional Lattice Diagram	Perkowski et al
Reed-Muller Tree	Reed-Muller Lattice Diagram	Perkowski et al
Kronecker Tree	Ordered Kronecker Lattice Diagram	Perkowski et al
Pseudo Reed-Muller Tree	Pseudo Reed-Muller Lattice Diagram	Perkowski et al
Pseudo Kronecker Tree	Pseudo Kronecker Lattice Diagram	Perkowski et al
Free Kronecker Tree	Free Kronecker Lattice Diagram	Perkowski et al
pD/nD Fixed/Mixed RM Tree	pD/nD Fixed/Mixed RM Lattice Diagram	

Table 3: *Relations of Canonical AND/EXOR Trees and Lattice Diagrams (PART I)*

Type of Tree	Lattice Diagram Generated from Tree and type of regular layout	Where lattices are discussed in more detail
Generalized Kronecker Tree (GKT)	Generalized Kronecker Lattice Diagram Lattice Diagram	similar, even more general diagrams are presented in Perkowski/Pierzchala/Drechsler
Pseudo Generalized Kronecker Tree (PGKT)	Pseudo Generalized Kronecker Lattice Diagram	Example 4.1 with canonical expansion, see also Perkowski/Pierzchala/Drechsler
Mixed Pseudo Generalized Kronecker Tree (MPGKT)	Mixed Pseudo Generalized Kronecker Lattice Diagram	
Ordered Generalized Kronecker Tree (OGKT)	Ordered Generalized Kronecker Lattice Diagram	
Free Generalized Kronecker Tree (FGKT)	Free Generalized Kronecker Lattice Diagram	
Linearly Independent Kronecker Tree	Linearly Independent Decision Lattice Diagram	Perkowski /Pierzchala/Drechsler
Boolean Ternary Decision Tree	Boolean Ternary Lattice Diagram	
Pair-Symmetry Ternary Decision Tree	Pair-Symmetry Ternary Regular Lattice Diagram	Example 4.2. - Non-canonical expansion, regular layout
Pair-Symmetry Ternary Decision Tree	Pair-Symmetry Ternary Generalized Lattice Diagram	Non-canonic expansion, non-regular layout

Table 4: *Relations of Canonical AND/EXOR Trees and Lattice Diagrams (PART II.)*

CANONICAL AND/EXOR DECISION DIAGRAMS

- Canonical AND/EXOR Decision Diagrams are obtained by combining **isomorphic** nodes and removing **redundant** nodes.
- For instance, a node with two inputs originating from the same node is removed (Drechsler).
- Similarly, a node with four inputs originating from the same node is removed.
- Transformations for S, pD and nD nodes are applied to single-variable nodes.
- Their multi-variable node generalizations are applied to multi-variable nodes.
- All these transformations generalize the OKFDD transformations.
- This way, the definitions of all flattened forms and decision diagrams can be obtained.

EXAMPLE WHEN PGKT IS BETTER THAN GKT

- The PGKT for fi has Shannon node for variable a on top, GRM node for cofactor $fi_{\overline{a}}$ for expansion variables b, c, and GRM node for cofactor fi_a for expansion variables b, c.
- The expansion diagram is

  $fi = \overline{a}(1 \cdot x \ \oplus \ b \cdot y \ \oplus \ c \cdot z \ \oplus \ bc \cdot v) \quad \oplus$
  $a(1 \cdot v \ \oplus \ \overline{c} \cdot y \ \oplus \ \overline{b} \cdot z \ \oplus \ \overline{b}\,\overline{c} \cdot x)$.
- Thus, it has two GRM nodes, and 5 nodes for variables a, x, y, z, v (the variable-nodes for variables x, y, z, v are shared).
- Observe that in the subdiagram for $fi_{\overline{a}}$ the polarity is [11,1,1] and in the subdiagram for fi_a the polarity is [00,0,0].

EXAMPLE WHEN PGKT IS BETTER THAN GKT (cont)

- It can be shown, that it is not possible to have a smaller diagram for $fi_{\overline{a}}$, because each change to another polarity would require exoring some of variables x, y, z, v and this would lead some extra exor nodes, which is to more nodes than four for only the variable-nodes.
- Similarly, changing the polarities in the subdiagram for a will always lead to diagrams with more nodes than in the above solution.
- Since both subdiagrams are worse, the entire diagram is worse by having the same expansion in both subdiagrams.
- Similarly it can be shown that the variable a must be on top, and, in general, there exists no GKT that is better or equal to fi.

CREATING NEW HIERARCHY

- Similarly it can be shown that:
- free GKT is better than the ordered GKT.
- ....
- all possible changes of orders in the subtrees to the same order in both subtrees will always result in higher total node costs.
- Examples of forms with special properties, like superiority of GKE with respect to PSDKRO, PGKE with respect to GKE, MPGKE with respect to PGKE, and so on, can be created.
- This way, the hierarchy from Fig. 4 has been also created.

APPLICATIONS

- Create very fast cube-based ESOP minimizers based on the idea of separating variables to **"Kronecker-type"** variables and **"GRM-type variables".**

- The concepts presented here can be applied for the synthesis of **easily testable** two-level AND/EXOR circuits.

- There are many **other new canonical expansion forms** resulting from the **flattening** of the new trees.

- Furthermore, variants of **non-ordered** sets S_j can also be considered.

APPLICATIONS (cont)

- Much research on **minimal ESOP circuits**.
- So far, the **exact solution** can be obtained only for a very small number of input variables.
- The quality of AND/EXOR circuits from these expansions should be better that those corresponding to GRMs or (Pseudo) KROs because the search space of GKTs is **much larger** than the search space of the GRM expansions.
- The size for which exact solutions can be found remains an opern problem.
- **Quality of search heuristics** to partition and to order the input variables.

ZHEGALKIN LATTICE DIAGRAMS AND REGULAR LATTICE LAYOUTS

- Akers introduced the so-called Universal Akers Arrays in 1972.
- They are regular lattices and look like BDDs for symmetric functions.
- It was proven that every binary function can be realized with such structure, but an exponential number of levels was necessary (which means, the control variables in diagonal buses were repeated very many times).
- Sometimes the only way to implement a function is to repeat the same variable **subsequently**, without other variables interspersed.

ZHEGALKIN LATTICE DIAGRAMS AND REGULAR LATTICE LAYOUTS (cont)

- The arrays of Akers were universal and they were unnecessarily large, because they were calculated once for all for the worst case functions.
- No efficient procedures for finding order of (repeated) variables were given, and some functions were next shown for which this approach is very inefficient.

LATTICE DIAGRAMS

- Because of the progress of hardware and software technologies since 1972, our approach is quite different from that of Akers.
- We do not want to design a universal array for **all** functions, because this would be very inefficient for **nearly all** functions.
- Instead we create a **logic/layout functions' generator** that gives efficient results for **many** real-life functions, not only symmetrical.
- As shown by Ross et al that, in contrast to the randomly generated "worst-case" functions, 98% of functions from real-life are decomposable.
- Therefore, the functions are either **decomposable** to the easy realizable functions, or they do not exist in practice.

LATTICE DIAGRAMS (cont)

- Analysis of realizations of arithmetic, symmetric, unate and standard benchmark functions and new technologies (Concurrent Logic, Atmel).
- Our generalizations **of expansions**:
  1. S,pD and nD expansions.
  2. All Linearly-Independent expansions, the Boolean Ternary expansions, all Zhegalkin expansions.
  3. All Kronecker, Pseudo-Kronecker, Mixed, and other Decision Diagram concepts that are used in Reed-Muller logic.

LATTICE DIAGRAMS (cont)

- More powerful **neighborhood geometries**.
- 2x2 Lattices, 3x3, 4x4 lattices.
- In essence, the 2x2,3x3,4x4 and 8x8 neighborhoods are used in patents and published works.
- We allow to mix control variables in diagonal buses. This permits to realize Free diagrams.

LATTICE DIAGRAMS (cont)

- Calculation of data input functions to lattice nodes for **any** type of expansions and **any** lattice neighborhoods is performed by the same technique of **solving logic equations** for a given structure, as one used for Linearly Independent logic.
- In contrast to LI logic, the structural equations can have **one, many, or no solutions**.
- When there are many solutions, the one evaluated as best is taken.
- When there are no solutions, the backtrack to another structure, another expansions, or another blocks of input variables is executed.

LATTICE DIAGRAMS (cont)

- Selection of the order of (usually repeated) variables is done using the concept of the best separation of most different-value minterms, using the Repeated Variable Maps.
- Variable ordering (repeating) and variable partitioning.
- In contrast to the worst-case randomly generated functions, for **real-life benchmark functions** only few repetitions of variables are enough.
- It is especially easy to **symmetricize** the weakly specified functions and Boolean relations.

CONCLUSIONS AND FUTURE WORK

The generalization of the DDs can be done in **several ways,** which can be **combined together** to create various new representations.

- D1. Creating new and **more powerful** expansion types for nodes (also non-canonical), thus departing from the standard S, pD, nD set of expansions.
- D2. Allowing for **several** types of expansions in every level of the expansion tree,
- D3. Allowing more **freedom in ordering variables** in branches of the tree (including the case of no ordering at all),

CONCLUSIONS AND FUTURE WORK. (cont)

- D4. Combining not only isomorphic nodes in trees to create Directed Acyclic Graphs but also **combining arbitrary nodes** together, thus modifying functions realized by these nodes. Nodes are combined using joining operations from [**?**, **?**, **?**]. This approach, in general, leads to the need of **repetitions of variables** (as in Regular and Generalized Lattices).
- D5. Creating **generalized expansions for sets of variables**, instead for single variables.

CONCLUSIONS AND FUTURE WORK. (cont)

- Some of these generalized types of decision diagram representations have already been introduced, investigated theoretically, and implemented in CAD tools to mention only those used for Exor Logic.
- **Many more**, however, remain still to be analysed and evaluated experimentally.
- Many types that may result from the above dimensions of generalization have not even been defined yet.
- Considering the Green/Sasao hierarchy, the new representations **will be not worse** than the known ones in terms of complexity.

CONCLUSIONS AND FUTURE WORK. (cont)

- We expect the new representations to find applications in logic synthesis for ESOP circuits, fine grain FPGAs, and representation of large functions.
- Symmetric functions.
- Zhegalkin Lattices are superior to Universal Akers Arrays when realizing all totally symmetric functions, partially symmetric functions, or easily symmetrizable functions for which only few variables require repetitions in the structure.
- They should be combined with **Ashenurst/Curtis decompositions**, for the layout-driven synthesisk the realization of **arbitrary** functions.

CONCLUSIONS AND FUTURE WORK. (cont)

- All these researches have applications to **submicron technology**
- They can be all used for custom logic/layout synthesis.
- To develop new types of FPGAs and synthesis for FPGAs.
- K*BMDs (Drechsler,Becker) use *arithmetic* operations of addition and multiplication instead of binary logic operations, and find applications in verification of digital systems, and for solving general problems in discrete mathematics.
- Here **another approach** to the generalization of Kronecker diagrams was proposed.
- All ideas here can be further extended to multiple-valued logic, integers, and word-level diagrams.

CONCLUSIONS AND FUTURE WORK. (cont)

- The diagrams and lattices introduced here can also be generalized to integer arithmetic (and also for rational arithmetic realized with continuous logic), where + and * arithmetic operators and more general linearly independent operations would be used.
- These *"word-level"* expansions can be derived.
- The word-level expansions together with the generalization types D1-D5 can be used to create trees, forms, diagrams, lattices, and layouts.
- Lattices have been also proposed for continuous logic and MV logic (Perkowski,Pierzchala,Drechsler).