

# Concurrent Decomposition of Multiterminal BDDs

Ilya Levin, Osnat Keren, Vladimir Ostrovsky,  
George Kolotov

Tel Aviv University, Bar Ilan University, Boston University

*The work is supported by BSF Grant 2002259  
"Optimizing Binary Decision Diagrams by using Autocorrelation  
Functions"*

# Outlook

- Decomposition of multi-output function of a large number of variables
- Parallel Connection of BDDs
- Existence of the disjunctive decomposition
- Definitions
  - D-polynomials
  - "Trapeze" implicant table
  - "Suitable" implicant tables
- Algorithm of the concurrent decomposition
- Experimental Results
- Conclusions

## Initial Considerations

- We deal with multi-output functions of a large number of input variables defined by a set of cubes of high order.
- Each of the cubes belongs to a specific characteristic function, which is a binary vector.
- We concentrate on functions having a large number of characteristic functions. It means that a number of cubes belonging to the same characteristic function is relatively small. In such cases using traditional methods of optimization is limited.

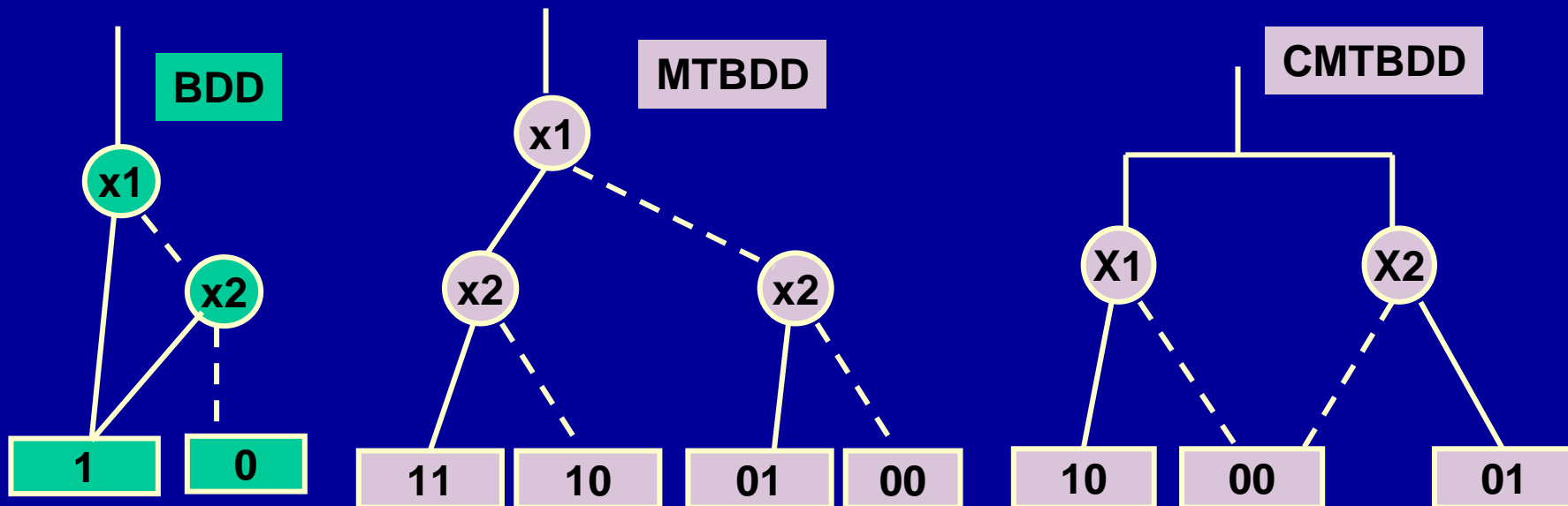
## Parallel connection of BDDs

- We introduce a parallel connection of classical BDDs
- The parallel connection corresponds to a disjunctive decomposition and to a specific product operation

# Simple Example

X1	X2	Y1
1	*	1
*	1	1

X1	X2	Y1	Y2
1	*	1	0
*	1	0	1

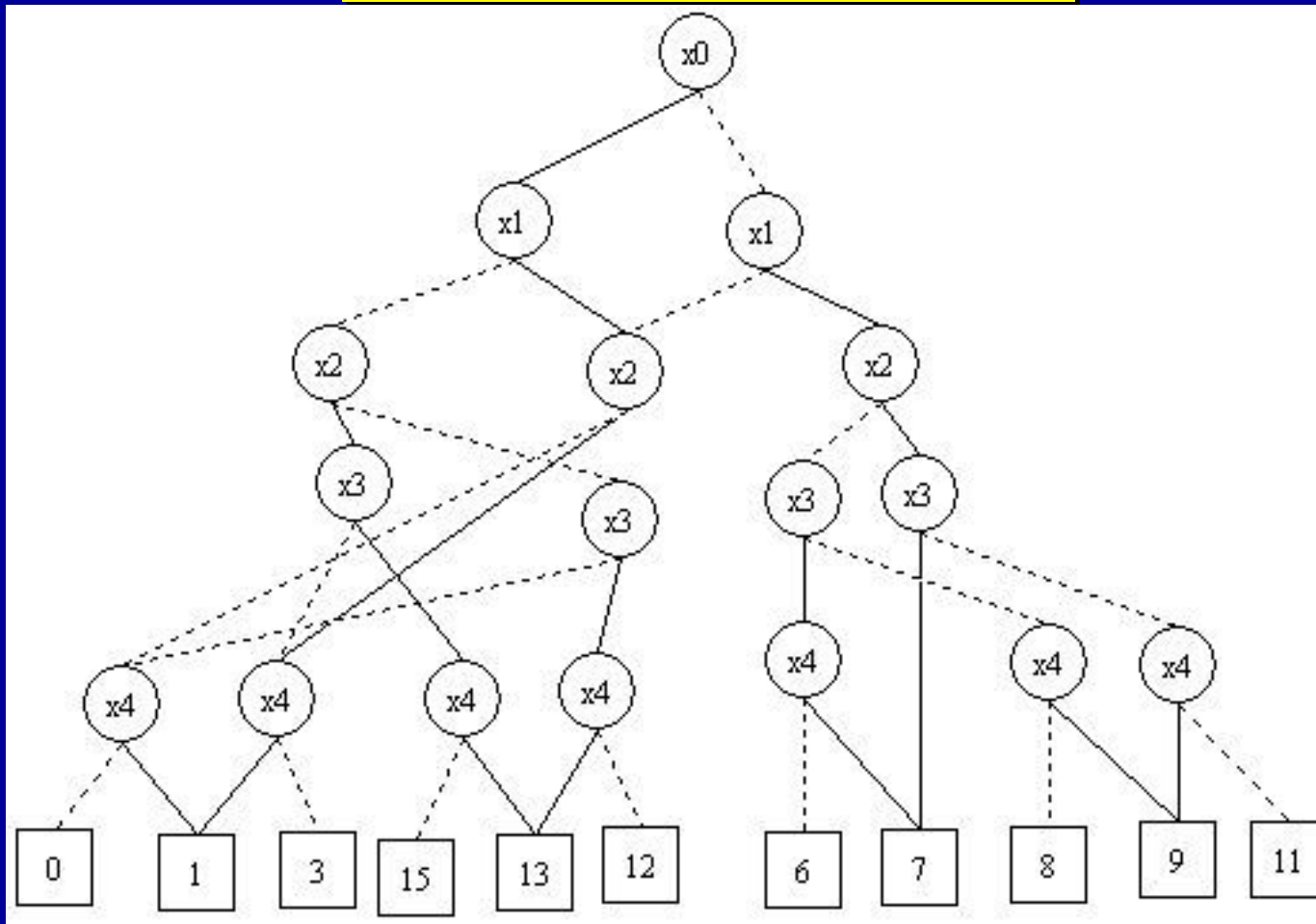


# Multi-output function defined by the list of cubes

5 input variables,  
4 output bits, 5 cubes,

#	X0	X1	X2	X3	X4	F0	F1	F2	F3
0	0	1	–	0	–	1	0	0	0
1	0	1	–	1	–	0	1	1	0
2	–	–	1	–	0	0	0	1	1
3	–	–	–	–	1	0	0	0	1
4	1	0	–	1	–	1	1	0	0

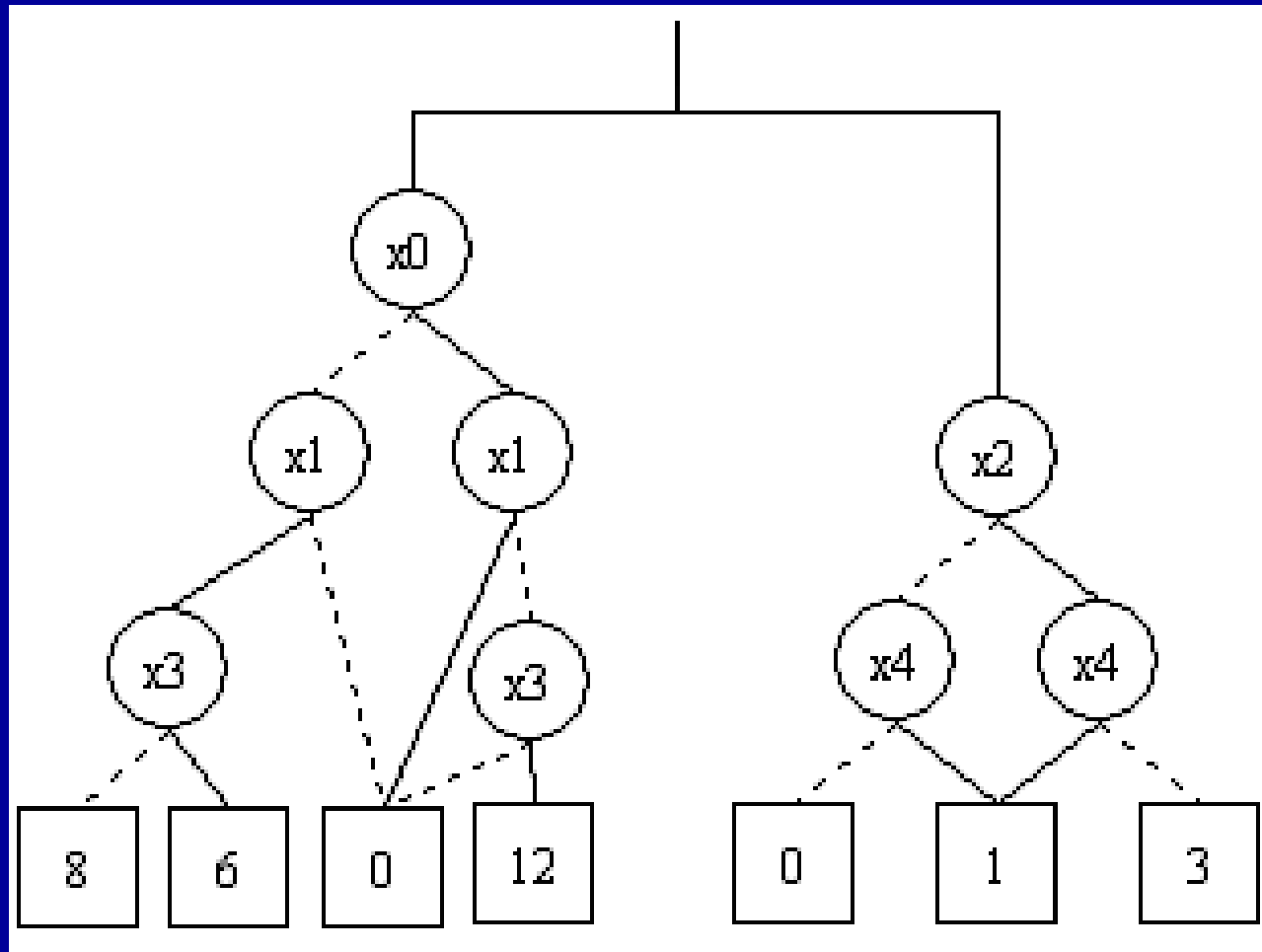
# Straightforward MTBDD implementation



**17 non-terminal nodes**

7<sup>th</sup> IWBP, September 21-22, 2006

# CMTBDD implementation



**8 non-terminal nodes**

7<sup>th</sup> IWBP, September 21-22, 2006

# Application to Planar BDD and Peace-wise Linearization

- Levin I., Stankovic R., Karpovsky M., Astola J., (2005) Construction of **Planar BDDs** by Using Linearization and Decomposition. *Proceedings of 14th International Workshop on Logic and Synthesis, Lake Arrowhead, California, pp. 132-139.*
- I. Levin, O. Keren, G. Kolotov, M. Karpovsky. (2006) **Peace-wise Linearization** of Multi-output Functions. *Proc. of the 2006 International Workshop on Spectral Methods and Multirate Signal Processing. Florence, Italy, pp. 345-353.*

## Disjunctive decomposition

Let  $y_i = f_i(x_1, \dots, x_n), i = 1, \dots, m$  be a system of logic functions defined by its implicant table  $X = \{x_1, \dots, x_n\}, Y = \{y_1, \dots, y_m\}$ .

We deal with a problem of disjunctive decomposition of this system, i.e. we consider a problem of representing the system in the following form:  $y_i = \varphi_i(X_1) + \psi(X_2)$ , where  $X_1, X_2 \subset X$ ,  $\varphi$  and  $\psi$  are implicant tables. Such a decomposition always exists in the case when  $X_1 = X$  or  $X_2 = X$ .

# Existence of a non-trivial decomposition

Theorem 1. The non-trivial disjoint decomposition of the function  $y_i = f_i(x_1, \dots, x_n), i = 1, \dots, m$  exists iff the function may be presented as an implicant table where each row includes "don't-cares" in each of a column belonging to one of the following sets of variables:  $X \setminus X_1$  or  $X \setminus X_2$ .

## *The goal of the Concurrent Decomposition*

- Beginning from the initial implicant table to construct a network of component concurrent BDDs
- Minimize each BDD independently
- Each of the components have to be "suitable" for the future implementation
- The suitability is defined as follows:

## *Suitable Table*

Definition. If a multi-output function defined by its implicant table may be translated into the MTBDD form by the straightforward Shannon expansion in such a way, that on any step of the expansion exists at least one variable without \* in all cubes, than such a table we call suitable table.

*The way to find the suitable fragments is based on the following Definitions*

# Cubes and Disjoint Cubes

Define function  $F(X)$  by a sum of cubes

$F(X) = \bigcup_1^m X^c$ . Each of the cubes  $X^c$  is

presented in a form  $X^c = \bigcap_{j=0}^{n-1} x_j^{c_j}$ , where:

$$x^w = \begin{cases} x & \text{if } w = 1 \\ \bar{x} & \text{if } w = 0 \\ 1 & \text{if } w = * \end{cases}$$

Cubes  $X^c$  and  $X^d$  are disjoint if:

$$X^c \cap X^d = 0.$$

# D-polynomials Representation of Disjoint Cubes

Definition.  $D$ -polynomial is a polynomial defined over a set of operators  $Y_i$

$$D = \sum_i Z_i Y_i + \alpha_0 Y_0 \text{ while the function } Z_i$$

satisfies the conditions:

a)  $\bigcup_i Z_i \vee \alpha_0 = 1$  (completeness),

b)  $Z_i \cdot Z_j = 0, \forall i \neq j$  (orthogonality).

$$D = \sum_i \left( \sum_{j \in I(i)} a_j \right) Y_i + \alpha_0 Y_0 = \sum_{j \in I(i)} a_{ij} Y_i + \alpha_0$$

where  $a_{ij}$  denotes  $j$ th -implicant of

function  $Z_i$ .

## Product of D-polynomials

Definition. Let  $D_1 = \sum_{i \in I(1)} a_{ij}^1 Y_i + a_0^1 Y_0$ ,  
and  $D_2 = \sum_{k \in I(2)} a_{kj}^2 Y_k + a_0^2 Y_0$ . The product of  
 $D_1$  and  $D_2$ , denoted as  $D_1 \circ D_2$  is defined  
by:  $D_1 \circ D_2 = \sum (a_{ij}^1 \times a_{kl}^2) \{Y_i \circ Y_k\}$ .

## Product of D-polynomials (cont')

$$D_1 \circ D_2 = \sum (a_{ij}^1 \times a_{kl}^2) \{Y_i \circ Y_k\},$$

over each pair of terms from  $D_1$  and  $D_2$ , including the implicit terms  $a_0^1 Y_0$  and  $a_0^2 Y_0$ .

Here  $a_{ij}^1 \times a_{kl}^2$  is a logic product (AND) of the corresponding  $a$ -functions and  $Y_i \circ Y_j$  is a combination of the respective operators. In other words: when  $a_{ij}^1 \times a_{kl}^2$  evaluates to 1, both  $Y_i$  and  $Y_k$  are computed concurrently.

# Superior Relation on the Set of Cubes

Definition. Cube  $X^i$  is a superior of cube  $X^j$  ( $X^i \succ X^j$ ) iff for any  $x^\omega \in X^i$  and  $x^\sigma \in X^j$  correct one of the following:

- 1)  $\omega \neq *$  **and**  $\sigma \neq *$  or
- 2)  $\omega \neq *$  **and**  $\sigma = *$  or
- 3)  $\omega = \sigma$ .

Cube  $X^j$  is an inferior of cube  $X^i$ .

## Example of the Superior Relation

Let function  $F(X)$  is defined as a set of cubes:

$$X^0 = \{13\} = x_0^1 x_1^0 x_2^1 x_3^1 = \cancel{x_0} \bar{\cancel{x_1}} \cancel{x_2} \cancel{x_3};$$

$$X^1 = \{6, 14\} = \bar{x}_0 x_1 x_2 \bar{x}_3 + \bar{x}_0 x_1 x_2 x_3 = \bar{\cancel{x_0}} \cancel{x_1} \cancel{x_2};$$

$$X^2 = \{0, 1, 2, 3\} = \bar{x}_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 + x_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_0 x_1 \bar{x}_2 \bar{x}_3 + x_0 x_1 \bar{x}_2 \bar{x}_3 = \bar{x}_2 \bar{x}_3 (\bar{x}_0 \bar{x}_1 + x_0 \bar{x}_1 + \bar{x}_0 x_1 + x_0 x_1) = \bar{\cancel{x_2}} \bar{\cancel{x_3}}$$

$$F(X) = \cancel{x_0} \bar{\cancel{x_1}} \cancel{x_2} \cancel{x_3} + \bar{\cancel{x_0}} \cancel{x_1} \cancel{x_2} + \bar{\cancel{x_2}} \bar{\cancel{x_3}}.$$

## Implicant table of the exemplary function

$x^c$	$x_0$	$x_1$	$x_2$	$x_3$	$y$
$x^1$	1	0	1	1	$y_1$
$x^2$	0	1	1	-	$y_2$
$x^3$	-	-	0	0	$y_3$

In the example:

$$\cancel{x^1} \succ \cancel{x^2}; \cancel{x^1} \succ \cancel{x^3}.$$

## Trapeze Tables

Definition. Let function  $F(X)$  be defined by its implicant table comprising a set of disjoint cubes  $X^c$ :

$$F(X) = \bigcup_1^m X^c, X^c = \bigcap_{j=0}^{n-1} x_j^{c_j} \text{ and}$$

$X^1 \succ \dots \succ X^m$ . If for each  $x_j^{c_j} \in X^c$   
 $\exists h: \forall l > h, l \in \{0, m\}; c_l = *$ , then the  
implicant table of function  $F(X)$  is  
called a trapeze table.

## Trapeze Table

If an multi-output function is defined by a set of disjoint cubes  $X^1, \dots, X^m$  ordered according to the *superior* relation:  $X^1 \succ \dots \succ X^m$  the corresponding implicant table is the *trapeze* function.

Example of the trapeze implicant table

$X^c$	$x_0$	$x_1$	$x_2$	$x_3$	$y$
$X^1$	1	0	1	0	$y_1$
$X^2$	0	1	1	-	$y_2$
$X^3$	1	1	-	-	$y_3$
$X^4$	0	0	-	-	$y_3$

## Quasi-trapeze Table

Definition. Let function  $F(X)$  be defined by its implicant table comprising a set of disjoint cubes  $X^c$ :  $F(X) = \bigcup_1^m X^c$ ,  $X^c = \bigcap_{j=0}^{n-1} x_j^{c_j}$ . If for each

$$x_j^{c_j} \in X^c \exists h: \forall l \leq h, l \in \{0, m\}; c_l \neq *$$

then the implicant table of function  $F(X)$  is called a quasi-trapeze table.

## Quasi-trapeze Table

- The set of the quasi-trapeze tables includes the trapeze.
- The implicant table the quasi-trapeze table can be implemented as a composition of the corresponding trapeze and a number of remained sub-tables.
- Such subtables we will call *tails*.

# Quasi-trapeze Implicant Table

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$y$
1	0	1	1	—	0	—	—	0	$y_1$
1	0	1	1	—	1	0	1	—	$y_2$
1	0	1	1	—	0	—	—	1	$y_3$
1	1	0	0	1	—	—	1	1	$y_4$
1	1	0	0	0	—	1	—	1	$y_5$
0	0	1	1	1	0	1	—	—	$y_6$
0	0	1	1	1	1	1	—	—	$y_7$
1	1	0	1	1	0	1	0	—	$y_8$
0	1	1	0	0	1	0	1	1	$y_9$

## Corresponding Trapeze Table

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$y$
1	0	1	1	—	—	—	—	—	$y_1$
1	0	1	1	—	—	—	—	—	$y_2$
1	0	1	1	—	—	—	—	—	$y_3$
1	1	0	0	1	—	—	—	—	$y_4$
1	1	0	0	0	—	—	—	—	$y_5$
0	0	1	1	1	0	1	—	—	$y_6$
0	0	1	1	1	1	1	—	—	$y_7$
1	1	0	1	1	0	1	0	—	$y_8$
0	1	1	0	0	1	0	1	1	$y_9$

# Corresponding Trapeze Table

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$y$
1	0	1	1	—	—	—	—	—	$y_1$
1	0	1	1	—	—	—	—	—	$y_2$
1	0	1	1	—	—	—	—	—	$y_3$
1	1	0	0	1	—	—	—	—	$y_4$
1	1	0	0	0	—	—	—	—	$y_5$
0	0	1	1	1	0	1	—	—	$y_6$
0	0	1	1	1	1	1	—	—	$y_7$
1	1	0	1	1	0	1	0	—	$y_8$
0	1	1	0	0	1	0	1	1	$y_9$

## Rectangle Fragment - Decoder

The *rectangle* fragments of any disjoint trapeze table comprises:

- a) a set of minterms of  $k$  variables; and
- b) a number of sets consisting of equal minterms of  $k$  variables.

MTBDD implementing the rectangle fragment of the implicant table will be called a *decoder*.

## ***Implicant Table of the Decoder***

$x_1$	$x_2$	$x_3$	$x_4$	$Y$
1	0	1	1	$F_1$
1	1	0	0	$F_2$
0	0	1	1	$F_3$
1	1	0	1	8
0	1	1	0	9

Prefixes 1101 and 0110 are parts of single cubes of the initial function. We call this prefixes as simple prefixes. Each of prefixes 1011, 1100 and 0011 are parts of sets of cubes. These prefixes we call *synthetic*. The synthetic prefixes define corresponding tails of the function.

# Tails

Tails  $F_1$ ,  $F_2$  and  $F_3$  are connected to the decoder's outputs 1011, 1100 and 0011 correspondingly

1011

$x_6$	$x_7$	$x_8$	$x_9$	$Y$
0	-	-	0	$Y_1$
1	0	1	-	$Y_2$
0	-	-	1	$Y_3$

1100

$x_5$	$x_7$	$x_8$	$x_9$	$Y$
1	-	1	1	$Y_4$
0	1	-	1	$Y_5$

0011

$x_5$	$x_6$	$x_7$	$Y$
1	0	1	$Y_6$
1	1	1	$Y_7$

7<sup>th</sup>

# Piecewise Ordered BDD

## Theorem 2.

Any multi-output function defined by its trapeze implicant table

a) is the suitable table;

b) may be implemented as an ordered MTBDD

## Theorem 3.

Any suitable table may be presented as a serial composition of trapezes.

## Block

Let  $F(X)$  be a multi-output logic function defined by its implicant table comprising an arbitrary set  $S = \{X^1, \dots, X^m\}$  of cubes  $F(X) = \bigcup_1^m X^c$ .

Select a subset  $\underline{S} \subseteq S$  such as:

a) cubes from  $\underline{S}$  are disjoint;

b) each of the cubes contains prefix  $X_{i(c)}^c \subseteq X^c$

satisfying the following relation:

$X_{i(1)}^1 \succ \dots \succ X_{i(m)}^m$ . The selected set is called block.

## Block Header and Tails

The selection of a block forms a trapeze implicant subtable of a specific multi-output function.

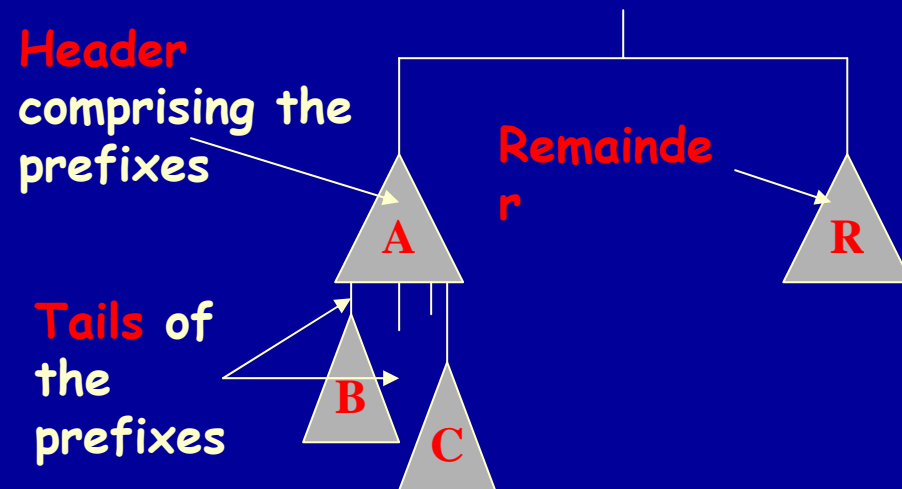
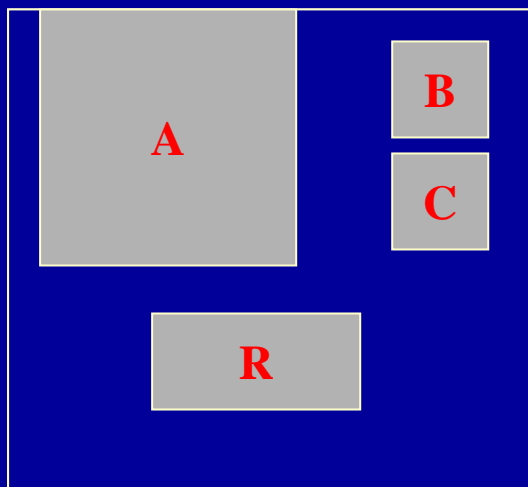
We will call this function a *header*. The header implements just a part of the block function.

The rest of the function may be implemented as a number of *tails* - sub-tables connected with corresponding outputs of the header.

Thus the block of function  $F(X)$  is implemented as a composition of the header and a number of corresponding tails.

# Concurrent Decomposition

- **A** is the **header** of the **block** is implemented as BDD and minimized
- **A**, **B** and **A**, **C** come from the same terms, thus **B** and **C** are **tails** of the block, implemented **in series** with the **header**
- **R** comes from the rest of terms - it is the **remainder** and will be implemented **in parallel**



# Concurrent decomposition algorithm

- Any implicant table of an arbitrary multi-output function can be decomposed into a net of linked trapeze tables.
- The constructing of the net is based on:
  - Partition of the initial table for a **block** - the pair-wise subset of disjoint cubes, and a **remainder** comprising all the rest of the initial table;
  - Decomposing the block into the **header** (a suitable table) and a number of **tails**;
  - Implementing recursively the above operations both for the remainder of the sub-table and for all tails of the selected block.

## *Criteria for Choosing the Basic Prefix*

- To divide variables between the block and the remainder
- To maximize the number of cubes disjoint to the basic prefix in the block
- To maximize the number of possible tails in the block

# Experiments with Standard Benchmarks

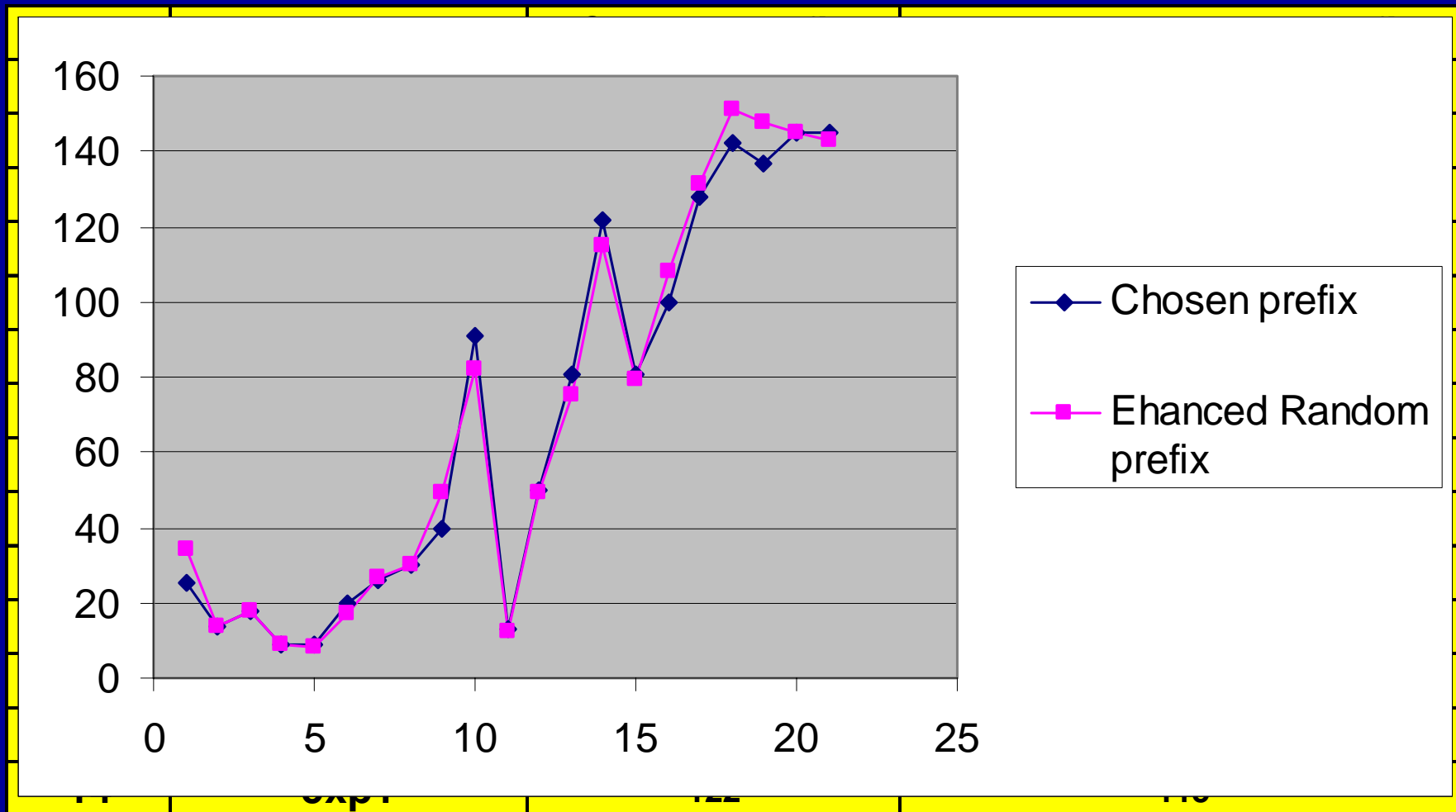
- Choosing the prefix by using the criteria (algorithm);
- Multiple Random search of the best prefix (Enhanced random prefix)
- Comparison according to:
  - Total number of nodes
  - Processor time

# Number of Nodes Chosen Prefix vs. Random Prefix

	Benchmarks	Chosen prefix	Random prefix
1	alu1	25	34
2	con1	14	14
3	f2	18	18
4	xor5	9	9
5	wim	9	8
6	dk27	20	17
7	misex1	26	27
8	squar5	30	30
9	inc	40	49
10	sqn	91	82
11	dc1	13	12

12	rd53	50	49
13	dc2	81	75
14	5xp1	122	115
15	sqr6	81	79
16	z4	100	108
17	root	128	131
18	adr4	142	151
19	radd	137	148
20	alu3	145	145
21	alu2	145	143

# Number of nodes

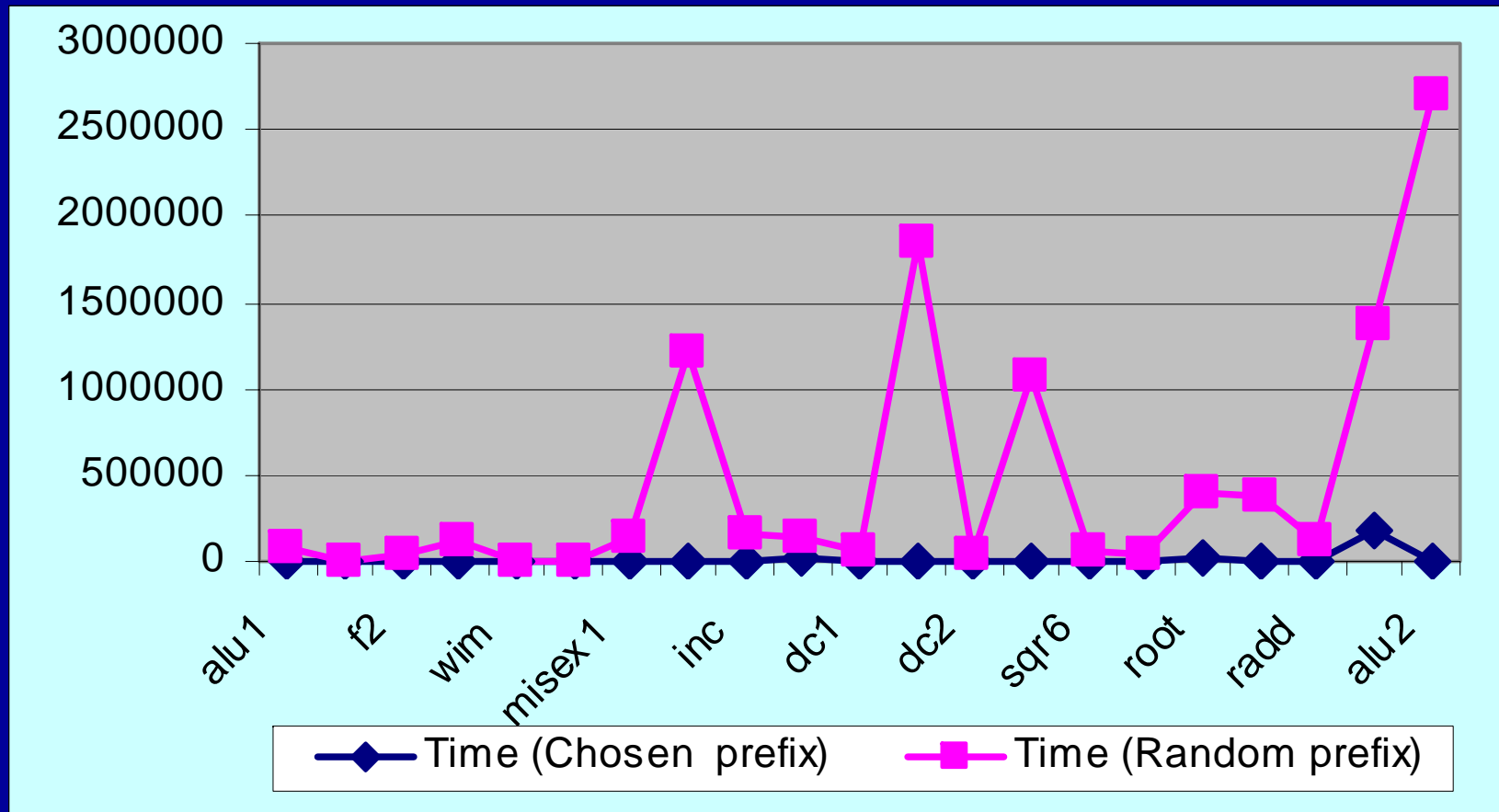


# Processor Time: Chosen Prefix vs. Random Prefix

Benchmarks	Time (Chosen prefix)	Time (Random prefix)
alu1	1234	88687
con1	1502	1462
f2	2420	45115
xor5	118	118120
wim	1150	4396
dk27	8680	3465
misex1	3005	129607
squar5	4506	1213595
inc	1618	168812
sqn	18637	132578

dc1	1732	65674
rd53	3535	1839635
dc2	6630	37281
5xp1	4930	1069359
sqr6	9293	56594
z4	5570	31828
root	14510	400578
adr4	6151	372586
radd	5602	120625
alu3	178390	1364522
alu2	4563	2699813

# Processor Time: Chosen prefix vs. Random prefix



# Conclusions

- A method for concurrent decomposition of multi-output functions of a large number of variables is presented
- A specific algebra of D-polynomials is introduced
- The proposed method provides significant reduction of BDD size
- There is a trade-off between a processor time and an efficiency of resulting BDD nodes
- The proposed algorithm allows to achieve quasi optimal results with minimal processor time