

# Split Multi-Terminal Binary Decision Diagrams

Ilya Levin, Osnat Keren

Tel Aviv University, Bar Ilan University, Israel

8th International Workshop on Boolean Problems September 18-19, 2008

# Outline

- ✓ Preliminaries
- ✓ Generalized ITE (GITE) operation
- ✓ D-polynomials
- ✓ Split MTBDD
- ✓ Dichotomy property
- ✓ Decomposition Algorithm
- ✓ Analysis of experimental results
- ✓ Conclusions

# Partitions

## **Definition.**

A partition on a set  $C$  is a collection of disjoint subsets of  $C$  whose set union is  $C$ , i.e.  $\pi = \{B_\alpha\}$  such that:

$$B_\alpha \cap B_\beta = \emptyset \ (\alpha \neq \beta) \text{ and } \cup \{B_\alpha\} = C.$$

Example:  $S = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$$\pi_1 = \{\{1\}, \{2\}, \{3, 4, 7\}, \{5, 6, 8\}\} = \{\bar{1}; \bar{2}; \overline{3, 4, 7}; \overline{5, 6, 8}\}$$

# Partitions

**A product**  $\pi_{prd} = \pi_1 \bullet \pi_2$  of partitions  $\pi_1$  and  $\pi_2$  is a partition comprising intersections of blocks  $\pi_1$  and  $\pi_2$ :

$$s \equiv t(\pi_1 \bullet \pi_2) \text{ iff } s \equiv t(\pi_2) \ \& \ s \equiv t(\pi_1).$$

**A sum**  $(\pi_1 + \pi_2)$  of the partitions  $\pi_1$  and  $\pi_2$  defined as follows:

$s \equiv t(\pi_1 + \pi_2)$  iff a chain  $s_0, s_1, \dots, s_n$  exists in  $C$  such as:

$s = s_0, s_1, \dots, s_n = t$ , for which either  $s_i \equiv s_{i+1}(\pi_1)$  or

$$s_i \equiv s_{i+1}(\pi_2), \quad 0 \leq i \leq n-1.$$

# Algebra of Partitions

The algebraic structure of partitions is known as a lattice.

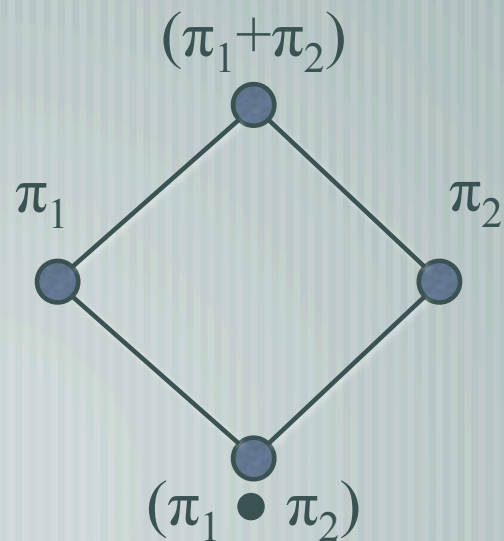
This lattice has both

Zero (the smallest partition  $\pi^0$ ) and

One (the biggest partition  $\pi^1$ ) elements defined as follows:

$$\pi^0 = \left\{ \overline{s_1}; \dots; \overline{s_m} \right\};$$

$$\pi^1 = \left\{ \overline{s_1, \dots, s_m} \right\}.$$



# Algebraic Decision Diagrams (ADD)

- ✓ Proposed in 1993 by R. Baher, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Parvo, F. Somenzi
- ✓ Multi Output Functions as ADD
- ✓ Different forms of representation of ADDs
- ✓ Operations: Apply and If-Then-Else operation
- ✓ Used for: matrix multiplication, shortest path algorithms, and numerical linear algebra.

# Algebraic Decision Diagram

An ADD is a function:

$$f : \{0,1\}^n \rightarrow S$$

where  $S$  is the finite carrier of the algebraic structure.

ADD is a form for representation of Multi Output Functions (MOF).

# Algebraic Decision Diagram

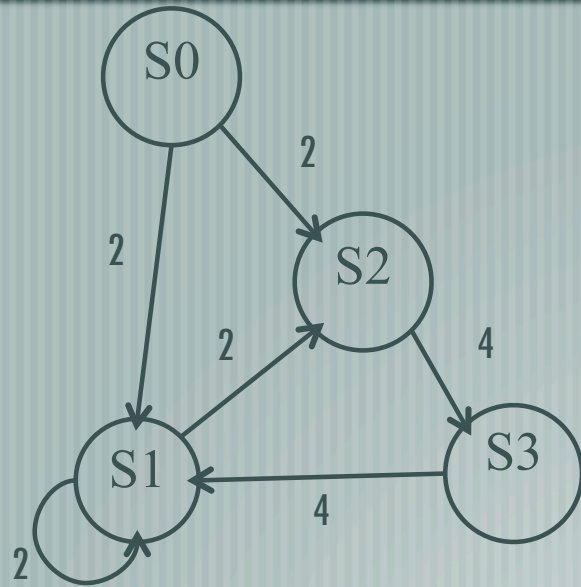
## ✓ ADDs representations

- Graph
- MTBDD
- Matrix

## ✓ ADD operations

- Apply
- If-Then-Else (ITE)

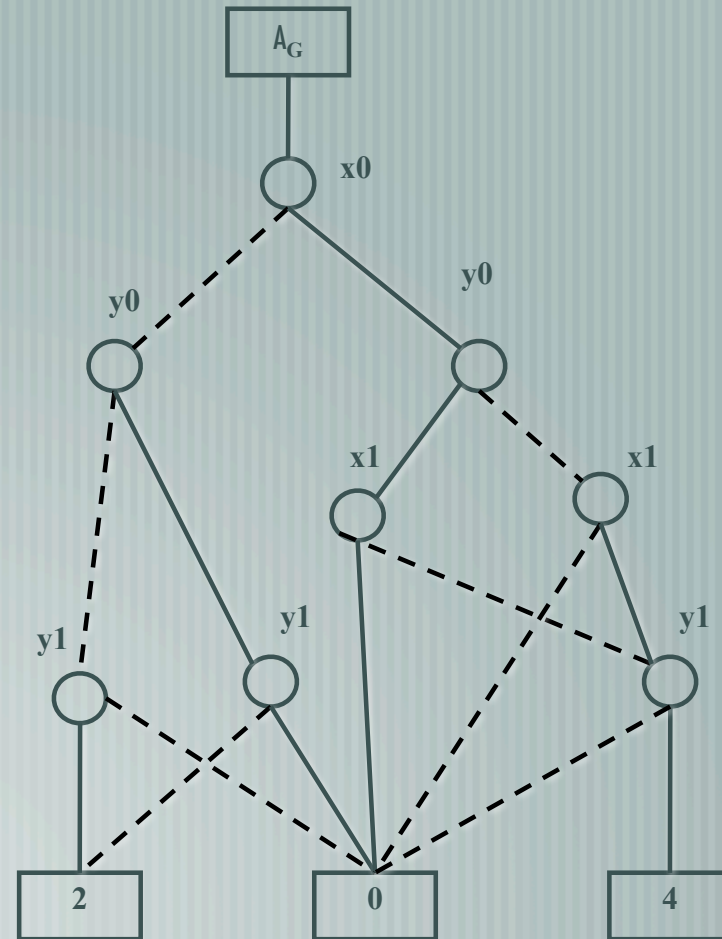
# Graph, matrix and MTBDD representations of ADD. Example (Baher, 1997)



$y_1, y_2$  00 01 10 11

$$\begin{pmatrix} 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 4 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$x_1, x_2$



# ADD Apply operation

$$\mathit{Apply}(f, g, op) = f \mathit{op} g$$

$$f = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}; \quad g = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{pmatrix}$$

$$\mathit{Apply}(f, g, +) = \begin{pmatrix} 5 & 5 & 4 & 4 \\ 5 & 5 & 4 & 4 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 3 \end{pmatrix}$$

# ADD If-Then-Else (ITE) operation

$$\mathbf{ITE}(f, g, h) = f \cdot g + \bar{f} \cdot h$$

$$f = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}; g = \begin{pmatrix} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{pmatrix}; h = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{pmatrix}$$

$$\mathbf{ITE}(f, g, h) = \begin{pmatrix} 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 3 \end{pmatrix}$$

The ADD ITE comprises a Boolean function operation as a “binary condition” being a two-block partition of the Boolean space

# Generalized ITE operation - GITE

- ✓ We define a new Generalized If-Then-Else (GITE) operation
- ✓ The GITE operation comprises a “multiple condition” that is *a partition* of the Boolean space
- ✓ The GITE allows to define algebra of MOFs
- ✓ Optimization of MOFs expressions can be formulated in terms of the GITE based algebra

# Generalized ITE operation

**Definition.** Generalized ITE (GITE) is

$$GITE(\pi, Y)$$

where:  $\pi = \{B_1; \dots; B_m\}$  is a partition on the Boolean space;

$Y = \{Y_1, \dots, Y_m\}$  – a set of operators.

$Y_i$  ( $i = 1, \dots, m$ ) corresponds to a certain block  $B_i$  of the partition  $\pi$ .

Thus, GITE consists of a partition portion and an operator portion.

# Multi Output Function

## Definition

*A Multi-Output Function (MOF)*

is a mapping  $f: \{0,1\}^n \rightarrow Y$ ,

which is **GITE**( $\pi, Y$ ) defined on two sets:

- a) the partitions  $\pi$  and
- b) the set  $Y$  of operators.

# MOF algebra

## **Definition.**

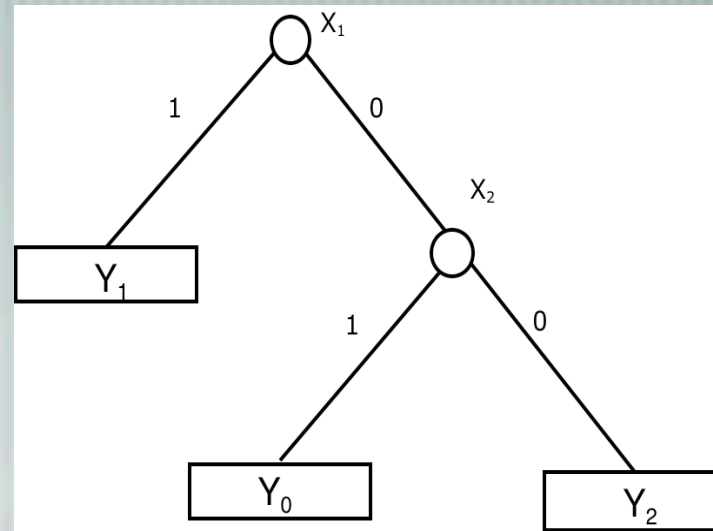
Algebra of MOFs is algebra formed by main sets

$\left( \{0,1\}^n ; Y \right)$  and  $**GITE**(\pi, Y)$  as the operation on these sets.

# Polynomial representation of MOFs

Example :  $D = B_1Y_1 + B_2Y_2 + B_0Y_0$

$$B_1 = x_1; B_2 = \bar{x}_1\bar{x}_2; B_0 = \overline{B_1 + B_2}$$



# D-polynomials

*D - polynomial* is a polynomial defined over a set of operators  $Y$ , as follows:

$$D = \sum_{i=1}^m B_i Y_i + B_0 Y_0 = \mathbf{GITE}(\pi, Y) = \mathbf{GITE}(B_1, \dots, B_m, B_0; Y_1, \dots, Y_m, Y_0)$$

where  $B_1, \dots, B_m$  - Boolean functions in a SOP form,

$$\pi = \{B_1, \dots, B_m, B_0\} \text{ and } B_0 = \overline{\sum_{i=1}^m B_i}$$

The *D*-polynomial is a specific case of the GITE operation

# MOF Apply operation

**Definition:** MOF Apply operation

$$\begin{aligned} \mathit{Apply}(D_a, D_b, op) &= D_a op D_b = \\ &= \mathit{GITE}(\pi_a \cdot \pi_b; Y_{a1} op Y_{b1}, Y_{a1} op Y_{b2} \dots, Y_{am} op Y_{bm}), \end{aligned}$$

MOF Apply operation is performed by multiplying partitions  $\pi$  and by pair-wise  $op$  operation on operators  $Y$ .

# Apply Operation on D-polynomials

**Definition.** Apply operation on D-polynomial we call *Product* of D-polynomials and define as follows:

$$\text{Let } D_1 = \sum_{i=1}^m Y_i + B_0^1 Y_0, \quad D_2 = \sum_{k=1}^l Y_k + B_0^2 Y_0.$$

$$D_1 \circ D_2 = \text{Apply}(D_1, D_2, op) = \sum \left( B_{ij}^1 \cdot B_{kl}^2 \right) \{ Y_i \text{op} Y_k \},$$

for each pair of terms from  $D_1$  and  $D_2$ ,

$B_{ij}^1 \cdot B_{kl}^2$  is a logic product (AND) of  $B$ -functions;

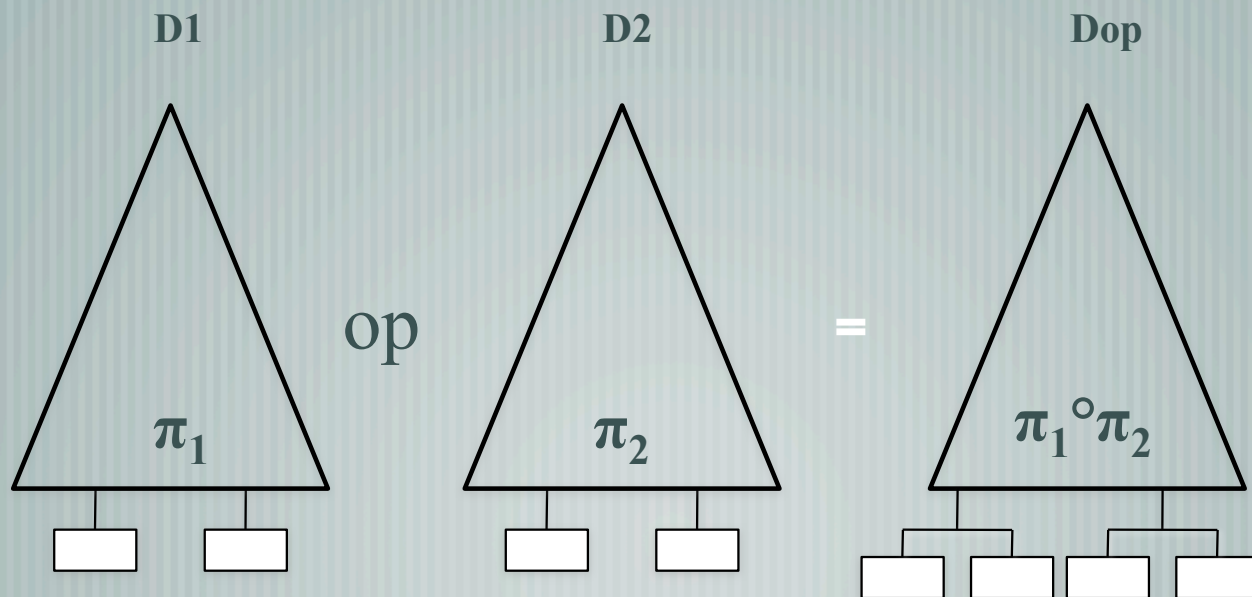
$Y_i \text{op} Y_j$  - is the Apply operation between  $Y_i$  and  $Y_k$

# MOF Apply operation

The Apply operation between  $D_1 = \mathbf{GITE}(\pi_1; Y_{11}, \dots, Y_{1m})$  and

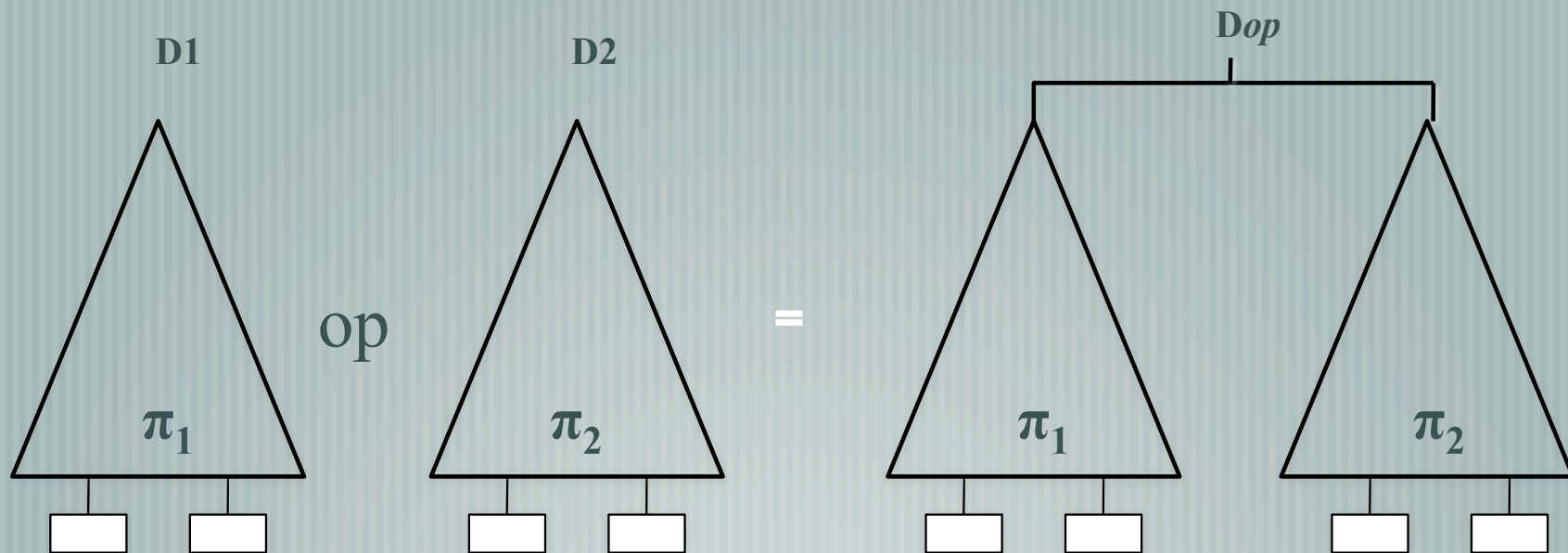
$D_2 = \mathbf{GITE}(\pi_2; Y_{21}, \dots, Y_{2m})$ :

$D_{op} = \mathbf{Apply}(D_1, D_2) = D_1 \mathit{op} D_2 = \mathbf{GITE}(\pi_1 \circ \pi_2; Y_{11} \mathit{op} Y_{21}, \dots, Y_{1m} \mathit{op} Y_{2m})$ .



# Alternative MOF Apply operation Apply

$$D_{op} = \mathit{Apply}(D_1, D_2) = D_1 \mathit{op} D_2 = \mathbf{GITE}(\pi^1; D_1 \mathit{op} D_2)$$



We call such a representation a *Split* representation

# Factorization of MOF expressions

The product of MOF partitions corresponds to the Apply operation,

The sum of MOF partitions corresponds to *factorization* of MOFs.

Define the factorization of MOFs as follows:

$$D = D_i \text{ op } D_j = \mathbf{GITE} \left( \pi_i + \pi_j; D_1, \dots, D_f \right),$$

where  $f$  is a number of blocks in  $(\pi_i + \pi_j)$

$D_1, \dots, D_f$  stand for MOFs representing remaining functions.

# Example

Let  $D_1 = x_1 Y_1 + \bar{x}_1 \bar{x}_2 Y_2 + \bar{x}_1 x_2 Y_3$ ,  $D_2 = \bar{x}_1 Y_4 + x_1 \bar{x}_3 Y_5 + x_1 x_3 Y_6$ .

$$\begin{aligned} D_1 \circ D_2 &= \mathbf{GITE}(\pi_1; Y_1, Y_2, Y_3) \circ \mathbf{GITE}(\pi_2; Y_4, Y_5, Y_6) = \\ &= \mathbf{GATE}(\pi_1 + \pi_2; D_{23}, D_{456}); \end{aligned}$$

$$D_1 \circ D_2 = D_3(D_{23}, D_{456}) = x_1 D_{23} + \bar{x}_1 D_{456};$$

where:

$$D_{23} = Y_1 \circ (\bar{x}_3 Y_5 + x_3 Y_6)$$

$$D_{456} = (\bar{x}_2 Y_2 + x_2 Y_3) \circ Y_4$$

# Substitution

Let  $D_1, D_2, D_3$  be MOFs :

$$D_1 = \mathbf{GITE}(\pi_1; Y_{11}, Y_{12}), D_2 = \mathbf{GITE}(\pi_2; Y_{21}, \dots, Y_{2m}),$$

$$D_3 = \mathbf{GITE}(\pi_3; Y_{31}, \dots, Y_{3m})$$

After substitution:  $Y_{11} \leftarrow D_2 \quad Y_{12} \leftarrow D_3$ , we have:

$$D_1 = \mathbf{GITE}(\pi_1; D_2, D_3)$$

# Split MTBDD

Split MTBDD can be constructed by connecting several MTBDDs in parallel (MOF *Apply*) and by *Substitution*:

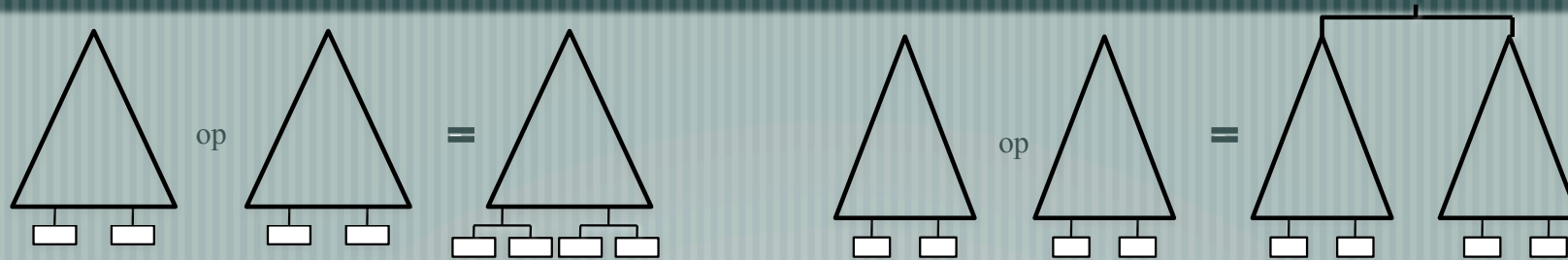
*Apply*: connecting roots of two or more MTBDDs.

*Substitution*: replacing a MTBDD terminal node with another MTBDD.

*D*-polynomials are analytical interpretation of the Split MTBDDs.

The parallel connection and the substitution of Split MTBDDs correspond to the product and the substitution of *D*-polynomials representing the Split MTBDDs.

# MTBDD Apply operation - Generalized Case



Two above MTBDD *Apply* operations are functionally equivalent and present two extreme solutions.

The optimal solution (according to a specific criterion) can be found between the two extreme cases.

In general, the *Apply* operation can be expressed as follows:

$$\begin{aligned}
 D_{op} &= \mathbf{Apply}(D_1, \dots, D_k) = D_1 \text{ op } D_2 \text{ op } \dots \text{ op } D_k = \\
 &= \mathbf{GITE} \left( \pi_{11} \cdot \dots \cdot \pi_{1h}; Y_{11} \text{ op } Y_{12} \cdot \dots \cdot Y_{1(h-1)} \text{ op } Y_{1h}; \mathbf{GITE} \left( \pi^1; D_{21} \text{ op } D_{22} \text{ op } \dots \text{ op } D_{2t} \right) \right), \\
 &(h+t=k)
 \end{aligned}$$

Finding of the optimal solution is the main goal of the proposed optimization method.

# Decomposition

- Beginning from the initial implicant table to construct a Split MTBDD consisting of a number of component MTBDD
- Minimize component independently
- Each of the components have to be dichotomic
- The dichotomy is defined as follows:

# Dichotomic Fragment

We say that a set of product terms forms a *dichotomic* fragment, if the set is straightforwardly mappable into an MTBDD.

The dichotomic property guarantees that there exists a Shannon expansion that will not bring additional product terms to the initial MOF.

The dichotomic property means that the paths of the MTBDD are in one-to-one correspondence with product terms of the MOF.

# Dichotomy Property

We assume that it is possible to formalize and implement some thinking templates comprising dichotomic fragments.

We study cases where a MOF is represented by a MTBDD.  
Our hypothesis is that the MOF can be more efficiently represented by a set of dichotomic fragments.

As a result, the whole MOF would be considered a set of sub-functions, the logical sum of which is equal to the output of the initial MOF.

Any MOF can be decomposed into a network of dichotomic fragments connected by the Apply and the Substitution operations.

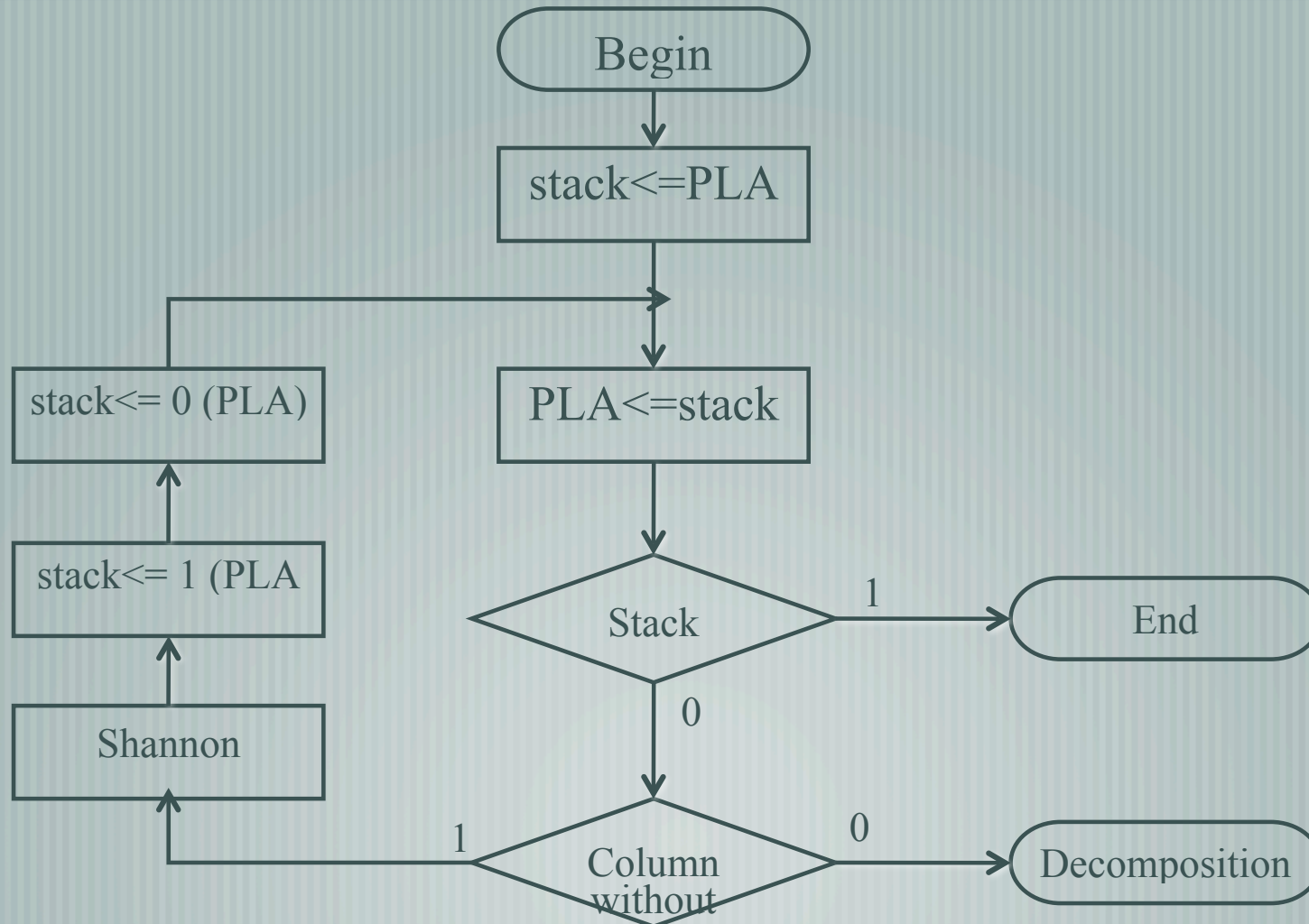
# General Decomposition Method

- ✓ Any implicant table of an arbitrary multi-output function can be decomposed into a net of linked dichotomic fragments.
- ✓ 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 (dichotomic fragment) and a number of "tails";
  - ✓ Implementing recursively the above operations both for the remainder of the sub-table and for all tails of the

# Two Algorithms of Decomposition

- ✓ Two algorithms have been developed and studied: a “density” algorithm and a “dichotomy” algorithm
- ✓ Both algorithms use one and the same general decomposition method
- ✓ The general method is the partitioning of the set of MOF cubes into a number of components. This partitioning is performed recursively
- ✓ On each step of the recursive procedure, the corresponding MOF component is partitioned into two subsets: a common header and a remainder
- ✓ Each common header is implemented as a conventional

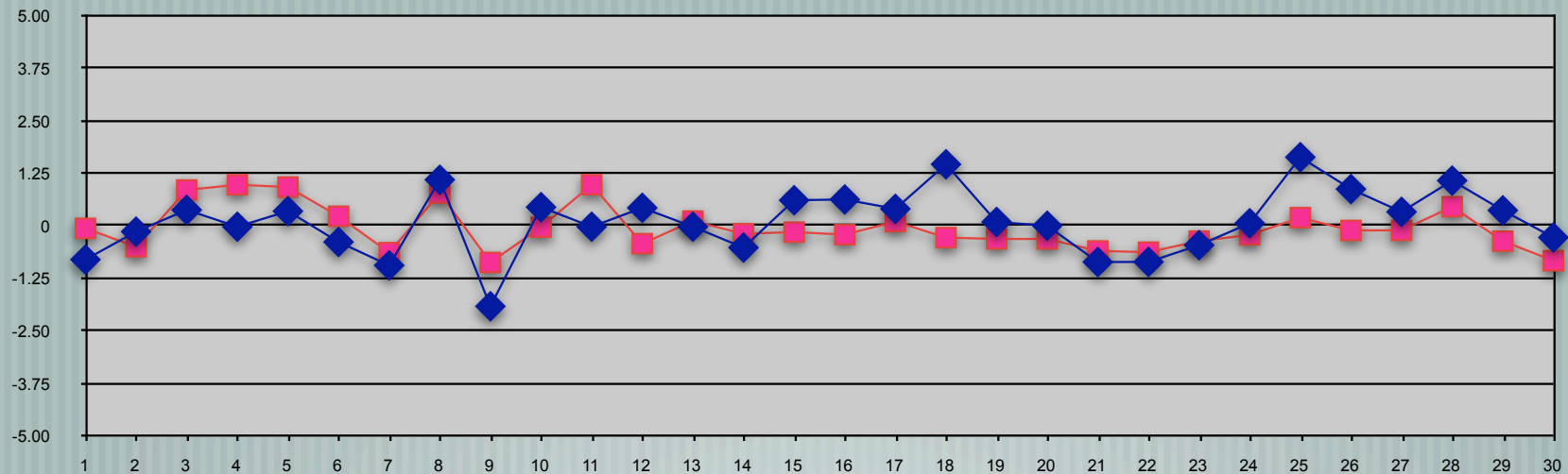
# Dichotomy Oriented Decomposition



# Experimental results

No	Benchmark	Inputs	Outputs	Products	Vectors	T/O	Non-DCs
1	dip	5	4	5	0	0	12
2	con1	7	2	9	2	0.222	23
3	f2	4	4	12	4	0.333	36
4	xor5	5	1	16	1	0.063	80
5	dc1	4	7	9	0	0	28
6	wim	4	7	9	0	0	18
7	dk27	9	9	10	0	0	31
8	rd53	5	3	32	3	0.094	144
9	alu1	12	8	19	7	0.368	41
10	sqn	7	3	38	6	0.158	184
11	sqar5	5	8	32	1	0.031	160
12	misex1	8	7	32	7	0.219	122
13	inc	7	9	34	10	0.294	189
14	dc2	8	7	40	8	0.2	213
15	z4	7	4	59	4	0.068	252
16	root	8	5	57	7	0.123	296
17	sqr6	6	12	50	8	0.16	202
18	9sym	9	1	87	1	0.011	522
19	adr4	8	5	75	5	0.067	340
20	radd	8	5	75	5	0.067	340
21	alu3	10	8	66	8	0.121	279
22	alu2	10	8	68	9	0.132	268
23	f51m	8	8	76	7	0.092	320
24	5xp1	7	10	75	9	0.12	296
25	rd73	7	3	141	3	0.021	840
26	dist	8	5	121	13	0.107	706

# Benchmark Results



Graph of correlation between:

1) MOF's density **Q** and

2) Difference **D** in the number of non-terminal nodes in an MTBDD and in the Split MTBDD

# Conclusions

- ✓ A Generalized ITE operation is introduced into the MOF algebra
- ✓ All operations of the MOF algebra can be expressed by the GITE operation
- ✓ A Split Multi Terminal BDD is proposed for a MOF representation
- ✓ The Split MTBDD is expressed by the GITE operation
- ✓ The Split MTBDD allows reducing the number of non-terminal nodes in comparison with the conventional MTBDD