

# Associative Digital Network Theory

*An Associative Algebra Approach  
to Logic, Arithmetic and State Machines*

NICO F. BENSCHOP

PART 2. Combinational Logic : *Associative, Commuting Idempotents*

Chapters 5 and 6 :

Symmetric and Planar Boolean functions  
as components of binary logic network synthesis,  
using rank spectrum analysis and phase assignment.  
Hamming- and Product codes for fault-tolerant logic :  
fast  $O(n^2)$  symmetric  $BF_n$  synthesis program 'Ortolog'  
scans all possible error correction codes to minimize Silicon.

**Chapter 5 :**

## Symmetric and Planar Boolean Logic Synthesis

### Copyright Notice :

Copyright (c) 2008 by dr. N.F.Benschop (author). Personal use of this material is permitted, but permission must be obtained from the author to reprint or republish this material in digital or hard copy form. Prior permission from the author is required to copy or otherwise republish, post on servers, redistribute to lists, or use any component of this work in other works for any purpose.

A fee may be charged for re-use. Abstracting with credit is permitted.

# Contents

<b>5</b>	<b>Symmetric and Planar Boolean Logic Synthesis</b>	<b>59</b>
5.1	Introduction . . . . .	59
5.2	Logic Synthesis independent of input ordering . . . . .	60
5.2.1	Orthogrid plot and rank spectrum . . . . .	60
5.2.2	Factoring paths by a planar node . . . . .	61
5.3	Symmetric and Threshold $BF's$ . . . . .	62
5.3.1	Symmetric functions 'count' . . . . .	62
5.3.2	$T$ -cell library , threshold logic cells . . . . .	63
5.4	Planar cut and factoring . . . . .	63
5.5	Fast symmetric synthesis: quadratic in nr. inputs . . . . .	64
5.6	Experiments and conclusion . . . . .	65
5.7	Planar Boolean logic synthesis . . . . .	65
5.7.1	All $BF_n$ are planar upto $n=4$ inputs . . . . .	66



## Chapter 5

# Symmetric and Planar Boolean Logic Synthesis

**Summary:** Described is the decomposition of any Boolean Function  $BF_n$  of  $n$  binary inputs into an optimal inverter coupled network of Symmetric Boolean functions  $SF_k$  ( $k \leq n$ ). Each  $SF$  component is implemented by Threshold Logic Cells, forming a complete and compact T-Cell Library. Optimal phase assignment of input polarities maximizes local symmetries. *Rank spectrum* is a new  $BF_n$  description independent of input ordering, obtained by mapping its minterms onto an orthogonal  $n \times n$  grid of (transistor-) switched conductive paths, minimizing crossings in the silicon plane. Using this ortho-grid structure for the layout of  $SF_k$  cells, without mapping to T-cells, yields better area efficiency, exploiting the maximal logic path sharing in  $SF$ 's. Results obtained with a *CAD* tool *Ortolog* based on these concepts, are reported. Relaxing Boolean symmetric functions to *planar*- functions covers a majority of Boolean functions, and improves synthesis, especially low-symmetry  $BF$  decomposition.

### 5.1 Introduction

Since the early nineteen eighties the synthesis of combinational logic for the design of integrated circuits (*IC*'s) is increasingly automated. Present logic synthesis, near the bottom of the *IC* design hierarchy just above layout, is fairly mature and is intensively applied in the design of production *IC*'s. But some problems remain:

- Logic synthesis tools often have a disturbing order dependence. Re-ordering signals, which should not affect the result, can cause a considerable increase or decrease of silicon area. To curb computer time, synthesis tools avoid global analysis which tends to grow exponentially with the number of inputs. Hence a local approach is preferred, with a greedy algorithm that takes the first improvement that comes along. The result then depends on the ordering of cubes in a *PLA* listing, or the input order in a *BDD* (binary decision diagram) [2] [3] representing a Boolean function ( $BF$ ). This effect is reduced by *global* analysis, and by symmetric function components  $SF$ , being independent of input ordering. *CPU* time is reduced by the 'arithmetization' via *rank spectrum* analysis of  $BF_n$ , a new method of characterizing  $BF$ 's, to be explained.
- Optimal choice of polarity, or phase assignment, of signals - either inputs or intermediate variables - is still an unsolved problem, although some heuristics are applied. The influence of input phases on logic symmetries will be exploited for an efficient  $BF$  decomposition.
- The use of a standard cell library is forcing decomposition- and cell mapping stages to produce

a sub-optimal gate network, versus *compiled cells* as needed [4] : not using a cell library but a programmable grid template, to be discussed. The proposed 'orthogrid' *BF* structure is an experiment in that direction, to be extended to planar *BF*'s beyond symmetric *BF*'s as grid template alternative to the known *FPGA* (field programmable gate array) or *FPMUX* (field programmable multiplexers) cells [5]. Performance prediction, which normally comes with a cell library, must then be provided by the cell compiler, which is quite feasible, replacing library maintenance by compiler support.

- Complete testing of combinational logic circuits requires *irredundancy*, guaranteed only in *sum of cubes* 2-level implementation. Logic in factored form, the usual result of a synthesis tool, sometimes has testability problems. Restriction to a *disjoint product* is proposed, with factors having no common input. This guarantees the irredundancy needed for *BF* testability in factored form. And: disjoint products yield a *spectral calculus*, with a *BF rank spectrum* independent of input ordering, and a convolution composition rule.

## 5.2 Logic Synthesis independent of input ordering

The mentioned problems in present synthesis *CAD* imply that no optimality (nor full testability) is guaranteed, nor does one know how close/far the optimum is. Presently, only by many synthesis runs (design space exploration) a feeling is obtained for the complexity of the functions to be synthesized, aiming for trading off circuit area, delay and power dissipation, but at a high cost in *CPU* time.

Our *aim* is to improve this situation, crucial for the future of digital *VLSI* systems. The emphasis is on order- independent function representation, using a spectral technique called *rank spectrum*, and on *global* analysis before synthesis, which then becomes feasible. In fact this is one step beyond *BDD* type of *BF* descriptions [3], by mapping minterms as paths in an orthogonal grid, using symmetric *BF*'s and signal phasing.

Then methods similar to those applied in signal processing, like the frequency spectrum, or convolution of impulse response and input sequence in the time domain, are applicable to Boolean functions. This yields:

- Logic synthesis employing global structure analysis  
by a rank-spectrum technique : 'arithmetization of Boolean algebra'.

### 5.2.1 Orthogrid plot and rank spectrum

Define Orthogrid plot as a mapping of each *minterm* (0/1 string of length  $n$ ) of a  $BF_n$  in an orthogonal grid, as an  $n$ -step path from the origin to the  $n$ -th diagonal. In input sequence, step down if '0', and right if '1' (see fig. 5.1). This models a network of pass transistors on silicon, with a conducting path from the origin to the  $n$ -th diagonal for the given minterm. *OR*-ing all paths yields function  $F=1$  only if some path connects origin to final diagonal.

For  $n$  inputs, each path ends on the  $n$ -th diagonal. All minterms of equal rank (number of ones) end in the same point on the  $n$ -th diagonal. Without confusion such minterm set is also called a *rank* of  $F$ . fig. 5.1 gives the orthogrid plot of a single rank *XOR* product (4 terms, rank 2)

Define a rank function  $RF$  to have only one non-empty rank.

The rank spectrum of  $BF_n$  is the vector of minterm counts per rank.

Hence a  $BF_n$  is the sum (disjoint union) of its rank functions.

Each rank has a number of minterms independent of input ordering, so the rank spectrum of any  $BF_n$  is also independent of input ordering.

In general, crossing paths in an orthogrid plot are not allowed to touch each other, to be drawn with a symbol for a bridge or tunnel. This makes the *orthogrid* style cumbersome to draw for larger functions, and probably explains the popularity of the well known (binary) Shannon expansion tree, which can be displayed free of crossings, thus as a planar a-cyclic graph. *Path sharing* is essential to recognize common factors, which is a clue to logic synthesis, showing the power of *BDD* and orthogrid representations.

### 5.2.2 Factoring paths by a planar node

*Def* : In an orthoplot a planar node connects all paths through it.

For instance the circled node in fig 5.1, and in fact all edge-nodes, which are of three types: I, T or L. All such paths are cut in two parts: each first section from the origin is continued (multiplied) by all second sections to the final diagonal.

A function  $F$  with all paths (minterms) passing through a planar node is a product  $F = G*H$  of two functions  $G(X)$  and  $H(Y)$  sharing no inputs, so  $X \cap Y = \emptyset$ , where  $G$  is a rank function; here  $G(a,b)$  and  $H(c,d)$ . A planar node plays the role of a *factor node*. *Planarization* is essential for synthesis, obtained by a proper choice of order and polarity of inputs.

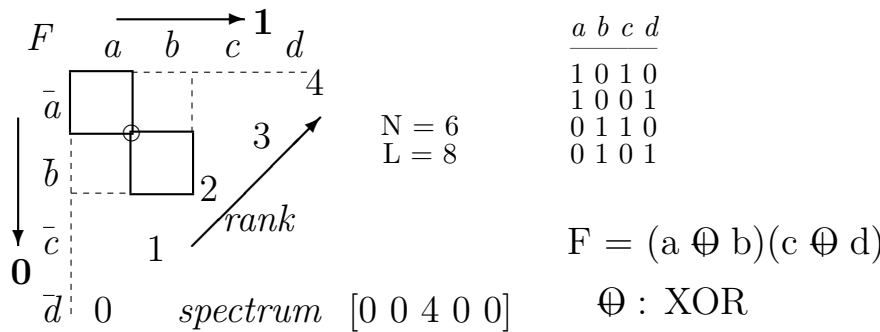


Figure 5.1: Gridplot of  $F = \text{XOR}$  pair product.

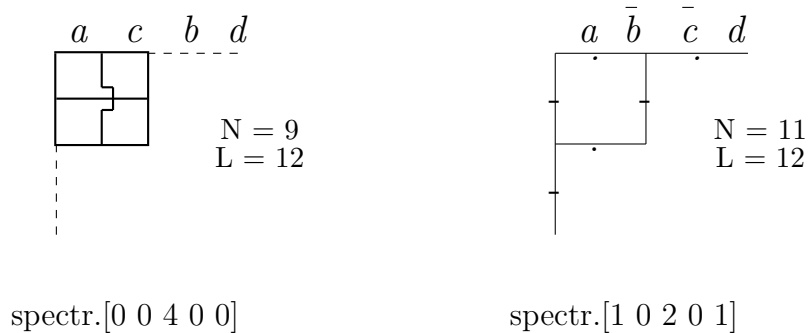


Figure 5.2: Planarize by permuting and/or inverting inputs.

Counting occupied gridpoints (nodes), multiple for non-planar nodes, yields a good factoring criterion for logic optimization (*planarization*) :

- Permute and invert (*phase*) inputs to minimize node count  $N$ .

Alternatively the number of links  $L$ , counting transistors, can be minimized. Node count  $N$  dominates over link count for practical technological reasons. Because a *bridge* requires two via's to another metal layer, costing more than a transistor which is simply a polysilicon line crossing (self-aligned) a diffusion path. In factored form fig. 5.1, permuting and inverting inputs yields the minimal  $(N, L) = (6, 8)$  values of the three gridplots of  $F$ .

This orthogrid representation allows characterization of special types of Boolean functions such as symmetric-, planar- and rank- functions, to be considered next. Notice the maximally  $2^n$  minterms are plotted in a square grid of  $n^2$  nodes, by virtue of dense *path sharing* as partial factors. Actually a half square suffices, up to diagonal  $n$ ; the other half plane could be used for the complement or dual of  $F$  (as in *CMOS*).

### 5.3 Symmetric and Threshold $BF$ 's

The well known Pascal Triangle, displayed in orthogonal grid fashion (fig. 5.3), gives in each node the number  $R(i, j)$  of all paths connecting that node to the origin. This is easily verified by its generation rule:  $R(i, j) = R(i - 1, j) + R(i, j - 1)$  is the sum of its predecessor node path counts. Induction yields the path counting rank spectrum.

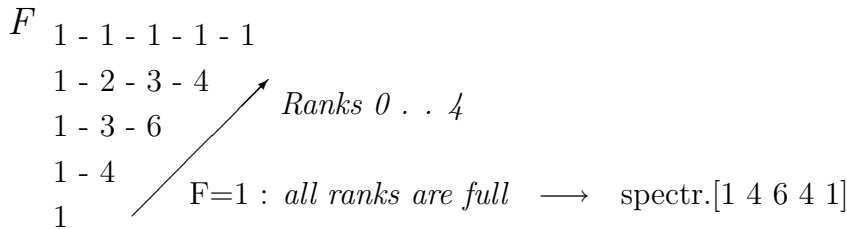


Figure 5.3: Binomial path-count for full ranks.

Clearly the *XOR*-product function  $F$  (fig. 5.1) is not symmetric in all inputs, but it has two partial symmetries or input equivalences (permute without changing  $F$ ), written  $a \cong b$  and  $c \cong d$ . Detecting and enhancing such partial symmetries is the basis of the 'ortolog' algorithm (section 5.5).

A rank=2 symmetric function in 4 inputs contains all minterms of rank 2, otherwise it cannot be an *SF*: there are  $(4 \text{ choose } 2) = 6$  minterms.

In fact a full rank has a *binomial coefficient* number of minterms.

Notice in fig. 5.1 there are two paths missing from a full rank=2 : 0011 and 1100 (see dotted lines), so  $F$  is not symmetric.

#### 5.3.1 Symmetric functions 'count'

Define a symmetric function  $SF$  as invariant for permuting its inputs.

Hence a function  $SF$  is symmetric in all inputs if it depends only on the *number* of high inputs in each minterm, and not on their ordering.

The ranks of a  $SF$  are either full or empty, so:

- A symmetric function  $SF[R]$  is determined by the set  $R$  of its *full ranks*, which is a subset of  $[0, .. ,n]$

An  $n$ -input function has  $n+1$  ranks, with  $2^{n+1}$  subsets, which is precisely the number of symmetric functions of  $n$  inputs. For instance the parity function is symmetric: it is active for each minterm with an odd number of 1-inputs. So the odd ranks are full, and all even ranks are empty:  $Parity(X) = SF[odd]$ , such as the *sum* function of a 3-input Full Adder.

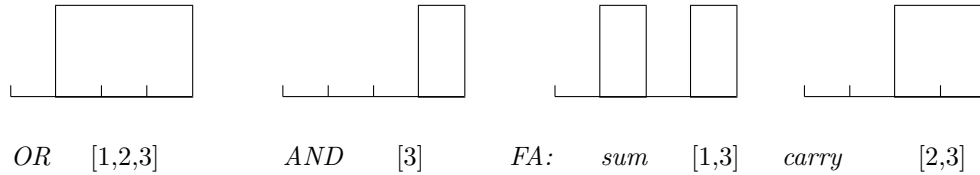


Figure 5.4: Rank plots of logic gates, and arithmetic Full Adder (FA).

Symmetric functions *count*, typical for arithmetic. The well known  $OR_n$  function of  $n$  inputs is symmetric, written  $SF_n[>0]$ : at least one high input, so only rank 0 is empty (see fig 5.4). The  $AND_n$  of  $n$  inputs function is  $SF_n[n]$ , thus active only if all  $n$  inputs are high, so only rank  $n$  is full (containing just one minterm). And in a 3-input Full-Adder ( $FA$ ): sum  $s = 1$  iff 1 or 3 inputs are high, so ranks  $[1,3]$  are full, written  $s = SF[1,3]$ . And the carry  $c = 1$  iff 2 or 3 inputs are high, so  $c = SF[2,3]$ .

Most  $BF$  however are not symmetric in all inputs, although many have partial symmetries (in some inputs). A factored function  $F$  cannot be symmetric, since inputs to different factors are not equivalent. So an  $SF$  has no factor, explaining why most logic synthesis tools, based on factoring, have trouble with efficient decomposition. This suggests putting  $SF$ 's in the Cell Library, with  $2^k$  cells  $SF_k$  of  $k$  inputs, halving the number of cells by using an inverter to exploit  $SF(-X) = -SF(X)$ .

### 5.3.2 $T$ -cell library , threshold logic cells

*Define* a threshold function  $T_k$  of  $n$  inputs, with a threshold  $1 \leq k \leq n$ ,  
then  $T_k=1$  whenever at least  $k$  inputs are active (high).

Any  $SF$  can be implemented by threshold logic functions  $TF$  as follows. The full ranks of an  $SF$  occur in intervals of successive ranks. Each interval  $[i, \dots, j-1]$  of full ranks is the  $AND$  of two threshold functions:  $T_i \cdot \overline{T_j}$ , so an  $SF$  with  $m$  fullrank intervals is the sum of  $m$   $TF$  pair products. For instance the FullAdder sum output (fig. 5.4) yields:

$$S[1,3] = (T_1 \cdot \overline{T_2}) + T_3, \text{ using the inverse of carry } T_2.$$

There are just  $n$   $TF$  functions of  $n$  inputs, with thresholds  $1, \dots, n$  - forming a compact and complete  $T$ -cell Library. Including an inverter, a T-cell library contains  $sum(1, \dots, n) = n(n+1)/2$  cells, that is 10 cells if  $n=4$ , or 15 cells for  $n=5$ . This is less than a complete S-cell library of  $1+(3+7+15)=26$  cells ( $n=4$ ), or 57 cells ( $n=5$ ), which however will yield more efficient synthesized circuits (see section on experiments).

## 5.4 Planar cut and factoring

The smallest asymmetric functions are:  $a(b+c)$ ,  $a+bc$  and  $\bar{a}b$ ,  $\bar{a}+b$ .

The first two cases use both  $(\cdot)$  and  $(+)$  where the role of  $a$  essentially differs from  $b, c$  which are equivalent (permutable). The last two cases are asymmetric in  $(a, b)$  but symmetric in  $(\bar{a}, b)$ . In general, input phasing costs little, making a function more symmetric and increasing local

symmetries (with dense path sharing), improving logic optimization (fig. 5.1, 5.2). In fact, it appears that:

the two basic causes for asymmetry are: *factoring* and *inverse*.

*Spectral product, and planar cut:* Function  $F = G(X) H(Y)$  is a *disjoint* product if factors  $G$  and  $H$  share no inputs, so  $X \cap Y$  is empty. Multiplying the rank spectra  $sp(G)$  and  $sp(H)$ , as a convolution, yields the spectrum of composition  $F = G.H$ :

$$sp(G_X.H_Y) = sp(G_X) * sp(H_Y) \quad (5.1)$$

Order input sets  $X$  and  $Y$  adjacent in the gridplot. Each path in  $G(X)$  is continued by (in series with) each path in  $H(Y)$ , thus forming all paths (minterms) of length  $|X| + |Y|$  in  $F$ , implying spectral product rule (5.1).

Let  $|X| = m$  then the gridplot of  $F$  has diagonal  $m$  consisting of only planar nodes, called a planar cut, with corresponding factor property. For example  $G = a \# b$  and  $H = c + d + e$  have spectra  $G[0, 2, 0]$  and  $H[0, 3, 3, 1]$  and  $m=2$ . Then product function  $F = G.H$  has as spectrum the product  $[0,3,3,1] \times [0,2,0] = [0,0,6,6,2,0]$  (longhand multiplication without carry).

## 5.5 Fast symmetric synthesis: quadratic in nr. inputs

The 'Ortolog' algorithm is designed for global yet fast detection of (partial) symmetries, enhancing them by input phasing. The rank spectrum is a simple and fast symmetry test for any sub function, by checking if each rank is full or empty.

The input format is that of a *PLA* (2-level *and/or* logic), hence a list of *cubes* as generalized minterms, each with all  $n$  circuit inputs (length  $n$  strings over 1/0/- for input straight or inverse or independent).

The algorithm is double recursive: start with a minimized 2-level logic  $BF_n(X)$  as a list of  $m$  cubes, and proceed as follows:

1. *Core(a, b)*: Collect all cubes symmetric in  $a, b$  for each input pair  $(a, b)$ . Maximize core by choosing input phase  $\bar{a}$  if  $Core(\bar{a}, b)$  has more cubes.
2. Input-expand maximal (phased) paircores to  $Core(a, b, Y)$  with inputs  $c$  (or  $\bar{c}$ ) in rest input set  $Y$ . Stop criterion:  $\max |Core| \times |inputs|^2$  prefers wide (more inputs) over deep Core (more cubes). Select one such 'best' multi input  $Core(Z)$ , symmetric for all inputs in  $Z \subseteq X$ . Let  $Y = \bar{Z} = X - Z$ .
3. Factorize  $Core(Z) = \sum_0^n G_r(Z) * H_r(Y)$  for ranks  $r \leq n$  with non-zero symmetric rank-functions  $G_r(Z)$  as factors (*planar cut*).
4. Recursively decompose (1-4) cofactors  $H_r$  until all components are symmetric.
5. Recursively decompose (1-5) remainder  $F(X) - Core(Z)$ , yielding an optimally phased network of symmetric functions coupled by inverters.

Speedup option: First partition  $F$  by collecting cubes with equal number of dont-cares (*DC* class), since cubes symmetric in the same subset of inputs likely have the same number of *DC*'s. Decompose the  $k$  subfunctions  $FDC_i$  separately, and *OR* them:  $F = \sum_1^k FDC_i$ .

The *SF* components can be implemented by *T*-cells, if a small *T*-cell library is preferred. However, *not decomposing* the *SF* cells yields better area efficiency, using their grid plot as layout pattern on silicon (*grid template*), maximally sharing logic paths.

The algorithm time complexity is  $O(n^2m)$ , for a  $BF_n$  list of  $m$  cubes with  $n$  inputs (step 1 is quadratic in  $n$ ). So only quadratic in the number of inputs (not exponential), and linear in the number of cubes. This allows *very fast synthesis* of many alternatives in a search for an optimal binary code at a higher level, as described in chapter 6: error correction codes in Boolean circuit design [6] [7] [8], or state-machine logic [9].

## 5.6 Experiments and conclusion

The described symmetric synthesis with a 15  $T$ -cells (max 5 inputs) library was compared with a known tool *Ambit* (Cadence) using a basic library of  $AND_n/OR_n/inv$  ( $n=2..5$ ) cells, or the usual extensive (full) library of several hundreds of cells. The logic *density* is the 0/1 filling % in the *PLA* table to be decomposed. Rather than number of cells, the total number of cell pitches is compared in Table 5.1 as area estimate.

cct	# inp	# cub	% density	Ambit	Ambit	Ortolog	Ratio
binom5	6	32	74	(126)	128	148	0.86
cordic	22	27	24	(135)	226	194	1.16
table3	14	52	75	(448)	718	902	0.80
parity	4	8	100	( 18)	41	48	0.85
Cell-Lib:				(full)	AOI	TC	AOI/TC

Table 5.1: Comparison of synthesis areas (Standard cell # pitches)

The threshold  $T$ -cell library is too restricted to compete with the usually very large cell libraries, since most  $BF_n$  do not have sizable local symmetries. Lacking special cells, such as  $XOR$  in the *parity* function, the area cost of  $T$ -cell mapping of  $SF$ 's is high.

The *Ortolog* algorithm performs *fast global* analysis, including phase assignment, of local symmetries in a  $BF_n$ . So one can employ it to detect, and enhance by input phasing, the (dense) symmetric parts of a circuit, for separate symmetric synthesis, with the remaining (sparse) asymmetric logic to be synthesized otherwise.

Moreover, its fast execution allows one to run it thousands of times within a few hours in order to optimize error correction code alternatives, with synthesized logic area as cost criterium, as reported in the next chapter.

Flexible *compiled cell* logic synthesis by a wider class of planar  $BF$ , derived from symmetric  $SF_n$  as *programmable* grid template, is described next.

## 5.7 Planar Boolean logic synthesis

The efficiency of decomposing to a network of symmetric Boolean functions clearly depends on the amount of (local) symmetries in the initial  $BF$ . Table 5.1 shows that restriction to a library of  $AND/OR$  (column *AOI*) resp. threshold  $T$ -cells (column *TC*) is too severe: results do not compete with the usually large cell library, except the *cordic* circuit which has much structure, viz. many local symmetries.

Symmetric components  $SF_k$ , with dense sharing of logic paths, should *not* be mapped onto  $T$ -cells, since the cost of decomposition is too high. Rather, they should be implemented directly as *planar* compiled grid cells.

*Def* : a planar Boolean function  $PF_n$  has a planar gridplot, possibly upon permuting and inverting inputs and/or inverting the output.

A symmetric  $SF_n$  is planar, having a gridplot with only planar nodes. . . . (5.2)

A symmetric  $SF_n$  can generate a set of planar functions is follows.

Let a link be a path of length=1 anywhere in a gridplot of some  $SF_n$ , in an *MOS* (metal oxide silicon) technology implementation corresponding to a transistor. Any  $SF_n$  is then a 'template' for a class of  $PF_n$  derived from it by removing one or more links. Conversely, any  $PF_n$  has a unique smallest covering  $SF_n$ , with the same set of non-empty ranks.

The class of  $PF$  is much larger than  $SF$ , while being easily derived by 'programming' (deleting links from) the  $SF$ 's as templates. The number of links in any  $SF_n$  is maximally  $\sum_1^n 2i = n(n+1)$ , hence in the order of  $|PF_n| = 2^{n(n+1)}$  planar functions of  $n$  inputs. Compared to  $|SF_n| = 2^{n+1}$  there are some  $2^n$  times more planar functions of  $n$  inputs than there are symmetric functions. A more precise estimate requires further investigation, regarding  $PF_n$  as component functions for logic synthesis.

### 5.7.1 All $BF_n$ are planar upto $n=4$ inputs

Define the 'structure' of a Boolean function to be:

invariant under some symmetry transformation. For instance:

- Permute inputs (some or all complemented) and/or complement output.

This defines the so called structural *pcio equivalence* between functions  $BF_n(X)$  of  $n$  inputs  $X = (x_1, \dots, x_n)$ , such as the 'dual'  $\overline{F(\overline{X})}$  of  $F(X)$ : complementing all inputs and the output preserves structure. For instance the Boolean 2-input logic gates  $AND(a, b)$  and  $OR(a, b)$  are each others dual, since by De Morgan's law  $\overline{a+b} = a.b$  and conversely  $\overline{a.b} = a+b$ , so complementing both inputs and the output transfers them into each other ('same structure' = isomorphic).

'Planarity' is a structural property of Boolean functions. Crossing of paths requires at least three inputs, so  $n \geq 3$  for non-trivial planar  $BF_n$ . In fact all  $BF_n$  with  $n \leq 4$  inputs are shown to be planar, thus each  $BF_n$  ( $n < 5$ ) is pcio equivalent to some  $n$ -input function with a planar ortho-plot, to be shown by complete inspection.

Each of the  $2^n$  input combinations (minterms) of a given  $BF_n$  is assigned a value true '1' or false '0'. So there are  $2^{(2^n)}$  Boolean functions of  $n$  inputs. By convention only the true minterms  $t_i$  are listed, as in fig 5.6, and a  $BF_n$  is the *OR* of its minterms in arbitrary order.

Define the weight of a  $BF_n$  as the number of minterms that make the function true, denoted  $W(BF_n) \leq 2^n$ .

Exclude the trivial true  $BF_n(X)=1$  and false  $BF_n(X)=0$  functions, and the functions that in fact depend on less than  $n$  inputs. Then there are only *six* pcio equivalence classes among the  $2^8$  functions in  $BF_3$ , and ... among the  $2^{16}$  of  $BF_4$ . This will be shown, using ortho-plot and spectrum, starting at  $n=1$ :  $BF_1(x)$  is  $x$  or  $\overline{x}$ , equivalent under input inversion, so there is one equivalence class for  $n=1$ .

**Lemma 5.1** *There are two pcio-equivalence classes of functions  $BF_2$ ,*

*equivalent to  $Z(+)$  and  $Z(\cdot) \text{ mod } 2$ , respectively  $XOR(a, b)$  and  $AND(a, b)$*

**Proof.** The full spectrum  $[1, 2, 1]$  of  $BF_2(a, b)=1$  covers the next non-trivial spectra of increasing weight:  $[0, 0, 1]$ ,  $[0, 2, 0]$ ,  $[0, 1, 1]$ ,  $[1, 1, 1]$  and  $[0, 2, 1]$ . However, the latter two have

complemented spectrum  $[0, 1, 0]$  resp.  $[1, 0, 0]$  which are pcio equivalent to  $[0, 0, 1]$ . Moreover  $[0, 1, 1]$  represents  $ab + a\bar{b} = a(b + \bar{b}) = a$  or  $ab + \bar{a}b = (a + \bar{a})b$  which do not depend on  $b$  resp.  $a$ , to be discarded as well. This leaves only two distinct pcio classes for  $n=2$ , with spectra  $[0, 2, 0]$  and  $[0, 0, 1]$  for Boolean expressions  $\bar{a}b + a\bar{b} = XOR(a, b)$  and  $ab = AND(a, b)$ . In fact they are the two basic arithmetic operations  $Z(+)$  and  $Z(\cdot)$  mod 2, represented as in fig 5.5.  $\square$

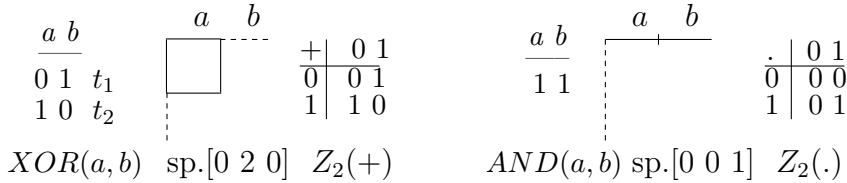


Figure 5.5: The two basic  $BF_2(a, b)$  equivalence classes:  $XOR_2$  and  $AND_2$ .

**Lemma 5.2** *There are seven pcio-equivalence classes of functions  $BF_3$ , all corresponding to planar functions.*

**Proof.** The proof is by inspection of spectra and corresponding equivalent functions. The full spectrum  $[1, 3, 3, 1]$  of  $BF_3(a, b, c)=1$  covers all 3-input spectra. Spectra of weights 5, 6 or 7 are obtained by function (output) complementation of  $BF_3$  with weights upto 4. Moreover, the spectrum of any  $BF_n$  is mirrored by inverting all inputs. Table 5.2 shows the five symmetric  $SF_3$ , with spectra [\*] having only empty or full ranks. The other non-equivalent  $BF_3$  spectra of *weight*  $\leq 4$  are derived from these. Write  $a - b$  for  $XOR(a, b)$ , and  $\cong$  for pcio-equivalence.

weight W	spectrum	functions of $W < 5$ in $BF_3$	pcio-eqv. spectra
1	$[0, 0, 0, 1]^*$	$abc$	all single term functions
2	$[1, 0, 0, 1]^*$	$abc + \bar{a}\bar{b}\bar{c}$	$[0, 1, 1, 0]$ by $c \rightarrow \bar{c}$
3	$[0, 0, 3, 0]^*$	2 high inputs, one low inp.	$[0, 2, 0, 1]$ by $c \rightarrow \bar{c}$
4	$[0, 0, 3, 1]^*$	2 or 3 high inps. ( <i>FA-carry</i> )	$[0, 2, 1, 1]$ by $c \rightarrow \bar{c}$
4	$[0, 3, 0, 1]^*$	1 or 3 high inps. ( <i>FA-sum</i> )	$[1, 0, 3, 0]$ by $c \rightarrow \bar{c}$
2	$[0, 0, 2, 0]$	$a(b - c)$	$[0, 1, 0, 1]$ by $b \rightarrow \bar{b}$
3	$[0, 0, 2, 1]$	$a(b + c)$	$[0, 1, 1, 1] \cong [0, 1, 2, 0]$

Table 5.2: The seven pcio-*eqv.*  $BF_3$  classes; five are symmetric  $SF_3$  [...]\*

All five non-*eqv.* spectra [\*] of symmetric functions  $SF_3$  are planar by (5.2). By inspection the remaining two non-*eqv.* spectra of  $BF_3$  correspond to planar functions. Moreover:

$[0, 0, 1, 1]$  of  $ab(c + \bar{c}) = ab$  is shifted full spectrum  $BF_1(c)$ : not in  $BF_3$

$[0, 1, 2, 1]$  of  $F(a, b, c) = a$  is shifted full spectrum  $BF_2(b, c)$ : not in  $BF_3$

$[0, 2, 2, 0]$  of  $XF_3$  has planar complement  $[1, 1, 1, 1] \rightarrow \overline{a - c}$  in  $BF_2$  (fig 5.6)  $\square$

The smallest case of a crossing is in  $BF_3$  (fig 5.6: terms  $t_1, t_3$  at  $b, c$ ). As an exercise to find non-planar functions, consider the worst case  $BF_n$  with a maximum number of crossings: the  $n$ -input maximal crossing function  $XF_n$  for each  $n > 2$ . Such function is unique, having one crossing at each non-edge inner node of its ortho-plot, thus  $\sum_{i=1}^{n-2} i$  crossings, with near-flat spectrum  $[0, 2, \dots, 2, 0]$  (figs 5.6-8).

Crossings are characterized by:

$$\text{A crossing node has only } [\dots, 0, 0, \dots] \text{ and } [\dots, 1, 1, \dots] \text{ term subsequences.} \quad (5.3)$$

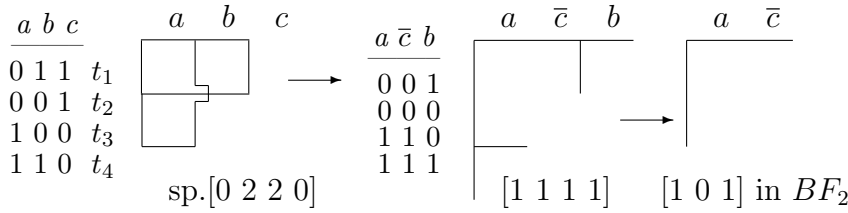


Figure 5.6: Max-crossing  $XF_3$  with one crossing (left), and planarized (right).

Including subsequences  $[.., 0, 1, ..]$   $[.., 1, 0, ..]$  yields a planar crossing. (5.4)

A node missing just one of these cannot be implemented, because any conducting contact between the two crossing paths of (5.3) implies both subsequences in (5.4), for brevity referred to as diagonal connections. As examples of planarization consider crossing functions  $XF_n$  for  $n \leq 5$ .

**Lemma 5.3 :**

$XF_{n>2}$  with a crossing at each inner node has a near flat spectrum  $[0, 2, \dots, 2, 0]$  and is planarized for  $n \leq 5$  by swap  $\bar{x}_n \longleftrightarrow x_2$  with spectra  $[1, 1, 1, 1]$  for  $n=3$ ,  $[1, 1, 2, 1, 1]$  for  $n=4$ , or  $[1, 1, 2, 2, 1, 1]$  for  $n=5$ .

**Proof.** By construction of  $XF_n$  the columns of first input  $x_1$  and inverted last input  $\bar{x}_n$  are identical. Permuting  $\bar{x}_n$  and  $x_2$  yields a planar ortho-plot only for  $n = 3, 4, 5$  – seen by inspection as follows.

Maximal crossing function  $XF_3(a, b, c)$  is given in fig 5.6: the crossing node is not planar due to missing terms  $[1, 0, 1]$  and  $[0, 1, 0]$ . The plot is planarized by permuting inverted last input  $\bar{c}$  and second input  $b$ , yielding a flat spectrum  $[1, 1, 1, 1]$ .

In general the spectrum varies with input inversions, but the sum total of minterms/paths is invariant.

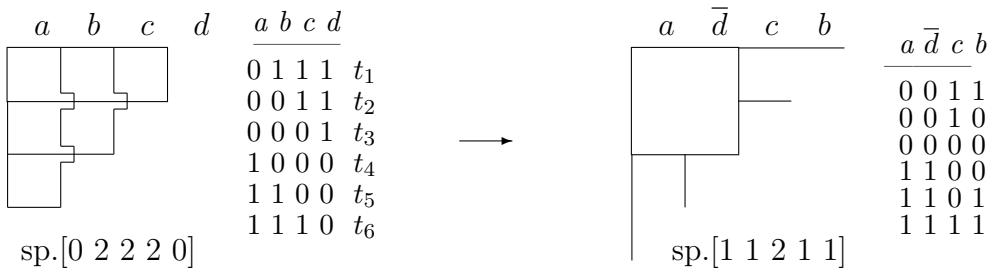


Figure 5.7: Maximal crossing  $XF_4$  with three crossings (left), and planarized (right)

Maximal crossing function  $XF_4$  has three crossings (fig 5.7), one at each inner node, with spectrum  $[0, 2, 2, 2, 0]$ . It is planarized by swapping  $\bar{d} \longleftrightarrow b$  which yields spectrum  $[1, 1, 2, 1, 1]$ . No other  $BF_4$  exists with a crossing at each inner node due to the absence of diagonal connections (5.3).

Maximal crossing function  $XF_5(a, b, c, d, e)$  (fig 5.8) has six crossings and spectrum  $[0, 2, 2, 2, 2, 0]$ . Permuting  $\bar{e}$  and  $b$  planarizes  $XF_5$ , with near flat spectrum  $[1, 1, 2, 2, 1, 1]$ . The crossing node at  $[0, 0, 1, 1]$  and  $[1, 1, 0, 0]$  is planar since all four subsequences of (5.3) and (5.4) are present.

The aim of input inversions is to minimize the Hamming distance  $D$  between subsets (pairs) of columns in the minterm representation of a  $BF_n$ . A permutation then clusters such minimal

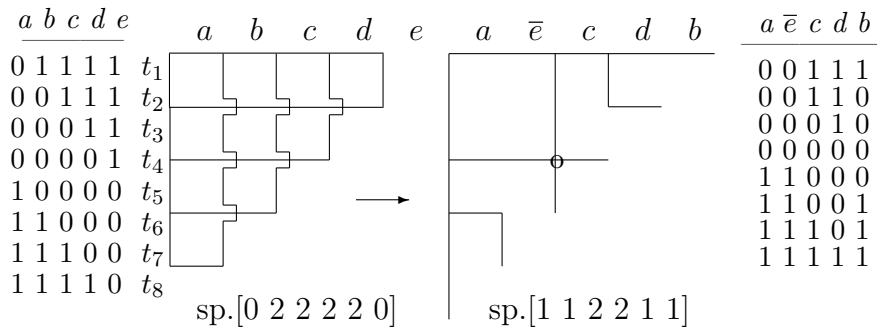


Figure 5.8:  $XF_5$  with 6 crossings (left), and planarized (right).

distance columns, and orders these clusters left-to-right with increasing mutual distance. For instance in  $XF_5$  (fig 5.8) the zero distance cluster  $D(a, \bar{e}) = 0$  is placed leftmost, followed by pair  $(c, d)$  with  $D(c, d) = 2$ . In fact, by the structure of  $XF_n$  holds  $D(x_i, x_{i+1}) = 2$  for  $1 \leq i < n$ , and pairwise clustering without inversions, except  $D(x_1, \bar{x}_n) = 0$ , is optimal.  $\square$

**Def:** A function  $F'(X, y) \in BF_{n+1}$  is an extension of  $F(X) \in BF_n$   
 if  $F'(X, 1) + F'(X, 0) = F(X)$ .

**Theorem 5.1** All Boolean functions  $BF_n$  for  $n \leq 4$  are planar.

**Proof.** A crossing requires at least 3 inputs, so by lemma 5.2 it suffices to show that each 4-input function in  $BF_4$  is a planar extension of some function in  $BF_3$ . Each planar representative of the 7 pcio-equivalent classes (table 5.2, column *spectrum*) is considered for initial non-planar extension, to be shown easily planarized.

Only functions with ranks 1 and 2 of value at least 2 can be extended in a non-planar fashion, to produce a crossing at node  $[0,0,1]$  and/or  $[1,1,0]$  as in fig 5.9. This excludes the first two functions ( $W=1, W=2$ ) leaving the remaining five functions to be considered.

Path extensions by extra input  $d$  are chosen such that two paths  $cd$  and  $\bar{c}\bar{d}$  form a non-planar crossing. It is readily verified (see fig 5.9) that this can in all five cases be planarized, for each of the chosen extension paths (dashed) at the remaining end-diagonal node 'o' of the considered  $BF_3$  function. Namely by using inverted inputs  $\bar{c}$  and  $\bar{d}$ . Moreover, in the middle case  $sp[0, 3, 0, 1]$  (Full Adder sum function) with extension spectra  $sp[0, 2, 1, 0, 1]$  and  $sp[0, 2, 1, 1, 0]$  the input pairs  $(a, b)$  and  $(\bar{c}, \bar{d})$  must be swapped as well.  $\square$

The smallest non-planar Boolean functions are in  $BF_5$ , thus at least 5 inputs, and they cannot be planarized by input permutations and/or input/output inversion(s).

For instance the function  $BF_5$  in fig 5.10 as non-planar extension of a  $BF_4$  function, where 'v' denotes a non-planar crossing, and '\*' a remaining non-planar node with two diagonal connections (5.4).

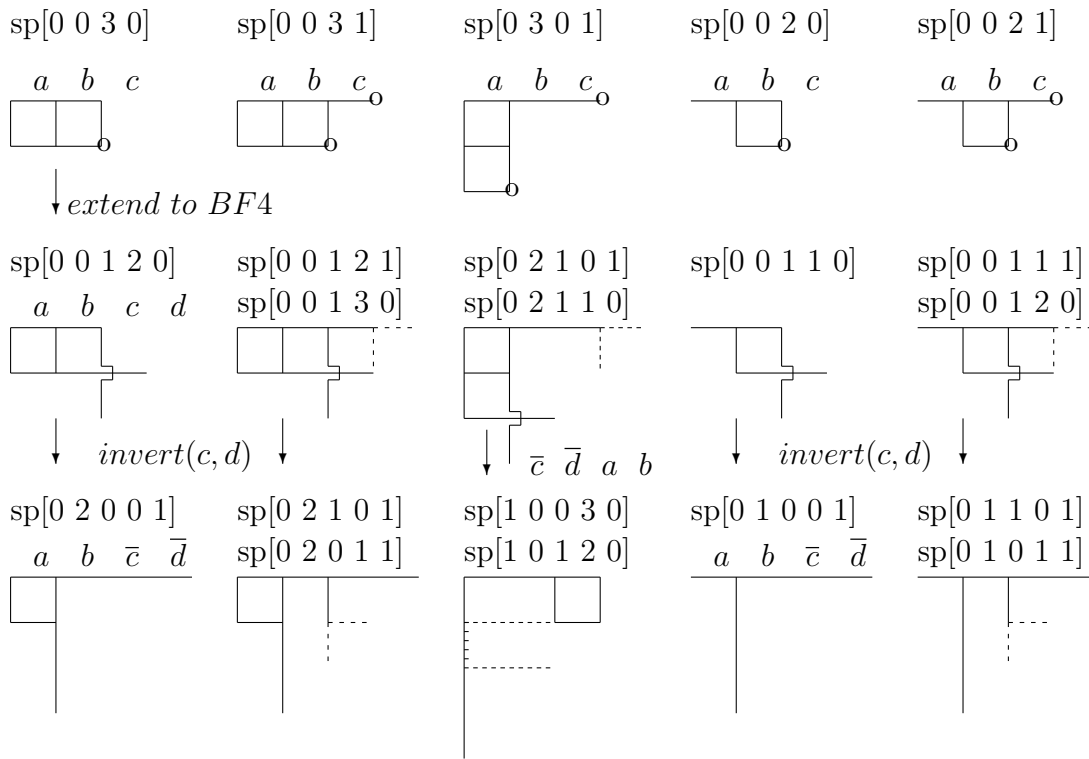


Figure 5.9: Planarization in  $BF_4$  of the five (table 5.2) extended  $BF_3$  functions.

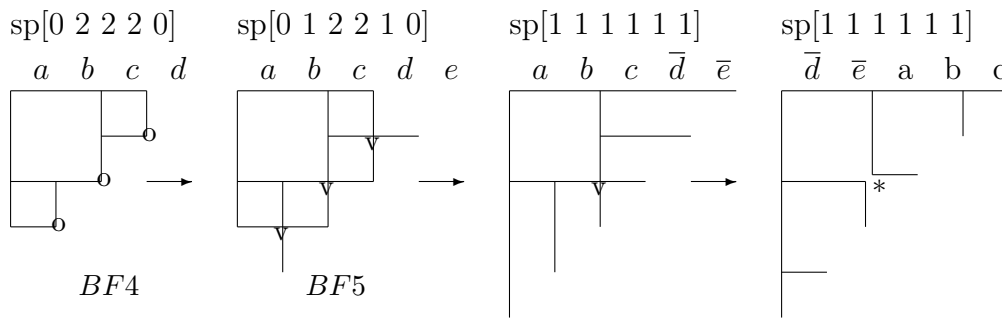


Figure 5.10: Non-planar  $BF_5$  function, as extension of a planar  $BF_4$  function.

## References

1. N.F.Benschop: "Symmetric Boolean Synthesis with Phase Assignment"  
*Information Theory Symposium*, U-Twente, Enschede, NL (may2001)
2. K. Akers: "Binary Decision Diagrams",  
*IEEE Comp. C-27*, 509-516 (1978)
3. R. Bryant: "Graph-based algorithms for Boolean function manipulation",  
*IEEE Comp. C-35*, 677-691 (1986).
4. J. van Eijndhoven – "CMOS cell generation for Logic Synthesis",  
*ASICON'94* 75-78, W.Y.Yuan (Ed), Beijing, 1994.
5. T. Courtney *et.al.*: "Multiplexer based reconfiguration for Virtex multipliers"  
*Field Progr. Logic and Appl's*, FPL2000 749-758, Villach, Austria (2000).
6. G. Muurling – *Fault tolerance in IC design using error correcting codes*,  
MSc thesis TU-Delft, NL. (2000)
7. G. Muurling *et.al.*: "Error correction for combinational logic circuits",  
*21-st Symp. on Info-Th.*, 25-31, Wassenaar, NL. (2000)
8. R. Kleihorst, *et.al.* – "Experiments with fault tolerant IC design using error correcting codes",  
*Int. Online Testing workshop*, Sicily. (2001)
9. N.F. Benschop: "Structure of Constant Rank State Machines",  
IFIP workshop *Logic and Architecture Synthesis*, 167-176, Paris, 1990.