Reversible Logic Synthesis with Garbage Bits

Lecture 6

Marek Perkowski

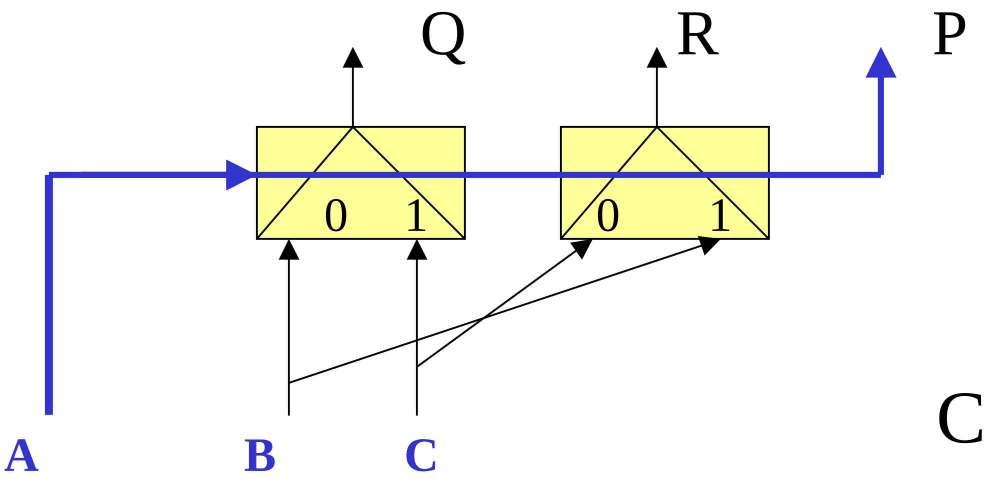
Logic Synthesis for Quantum Pseudo-Binary Logic (Permutation Logic)

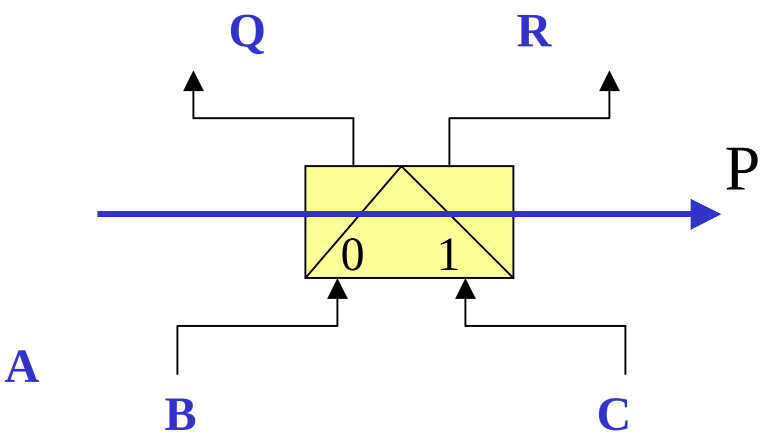
Background

Reviev of Reversible logic

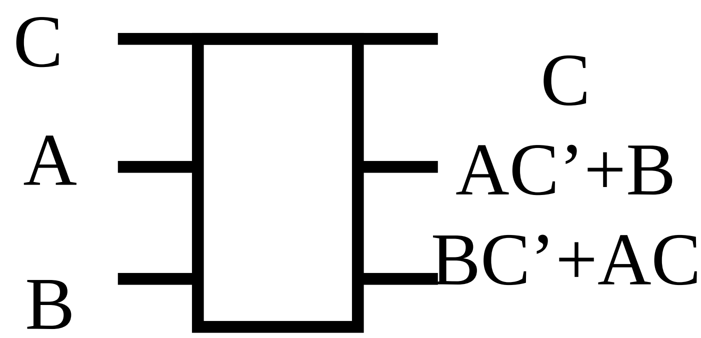
This approach is mostly for quantum logic realization

For optical and CMOS realizations the k*k assumption is not necessarily used





A circuit from two multiplexers

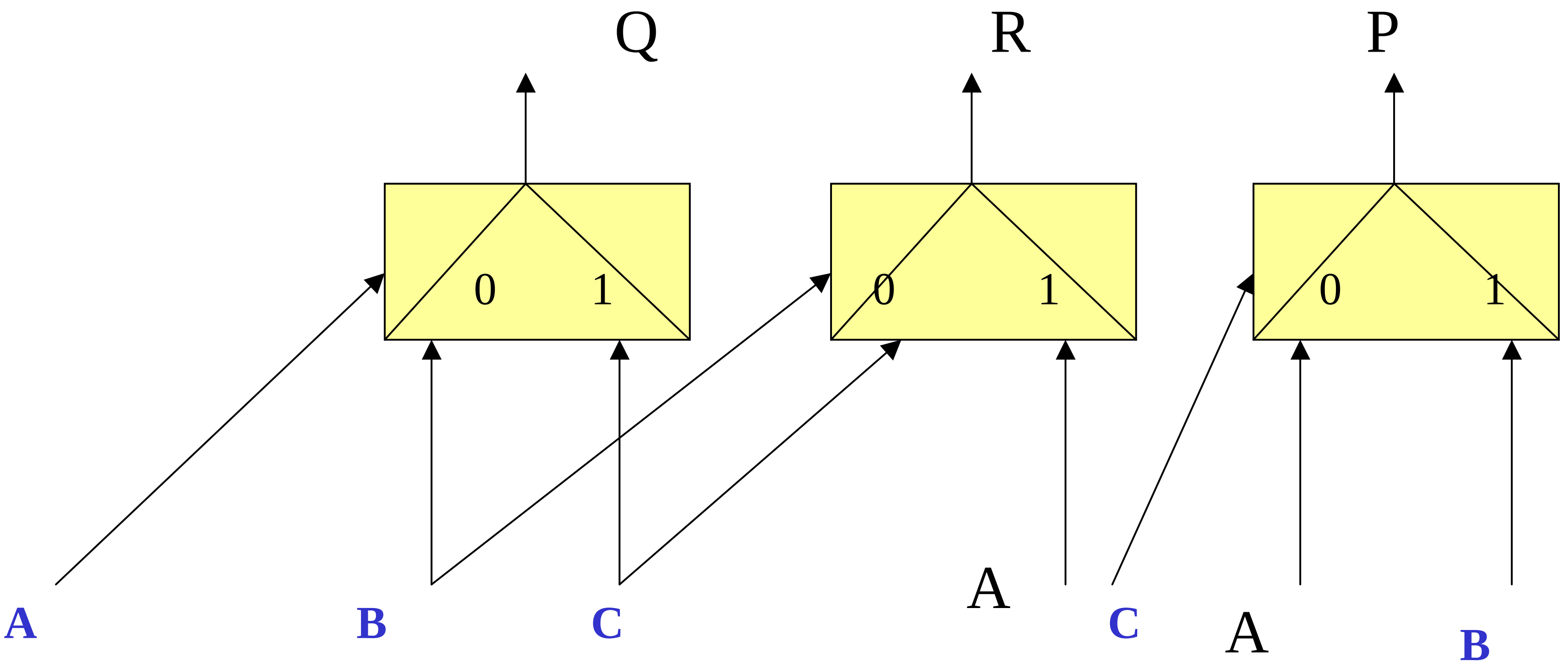


Its schemata

This is a reversible gate, one of many

Notation for Fredkin Gates

Margolus Gate



Reversible

Conservative

3*3 gate

Toffoli Gate

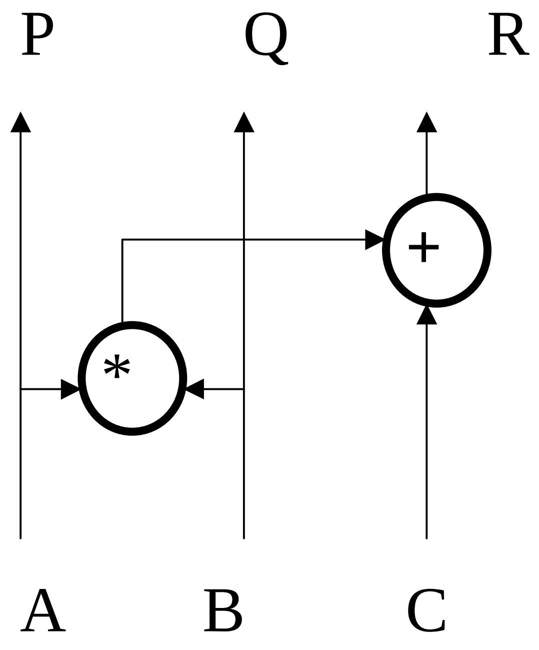
- The 3 * 3 Toffoli gate is described by these equations:

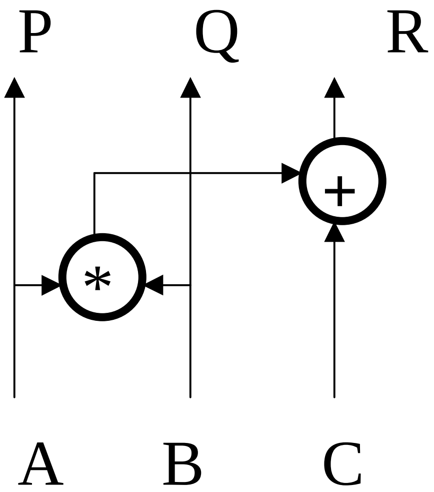
  $\boldsymbol{P = A,}$

  $\boldsymbol{Q = B,}$

  $\boldsymbol{R = AB \oplus C,}$

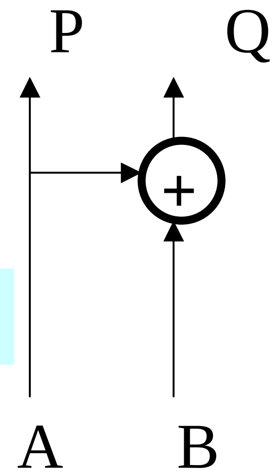
- Toffoli gate is an example of ***two-through*** gates, because two of its inputs are given to the output.



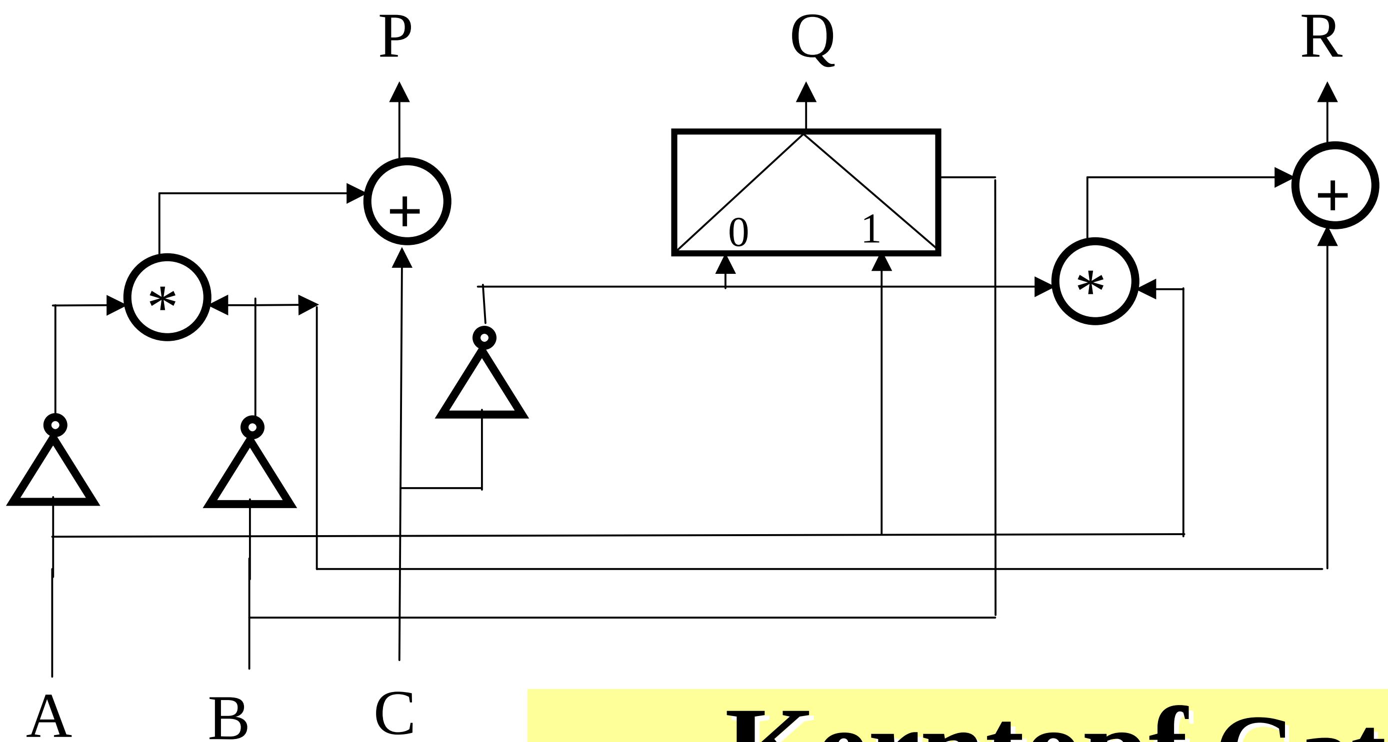


Toffoli

Feynman



Feynman, Toffoli and Fredkin gates are their own inverses



Kerntopf Gate

Kerntopf Gate

- The Kerntopf gate is described by equations:

  $P = 1 \oplus A \oplus B \oplus C \oplus AB,$

  $Q = 1 \oplus AB \oplus B \oplus C \oplus BC,$

  $R = 1 \oplus A \oplus B \oplus AC.$

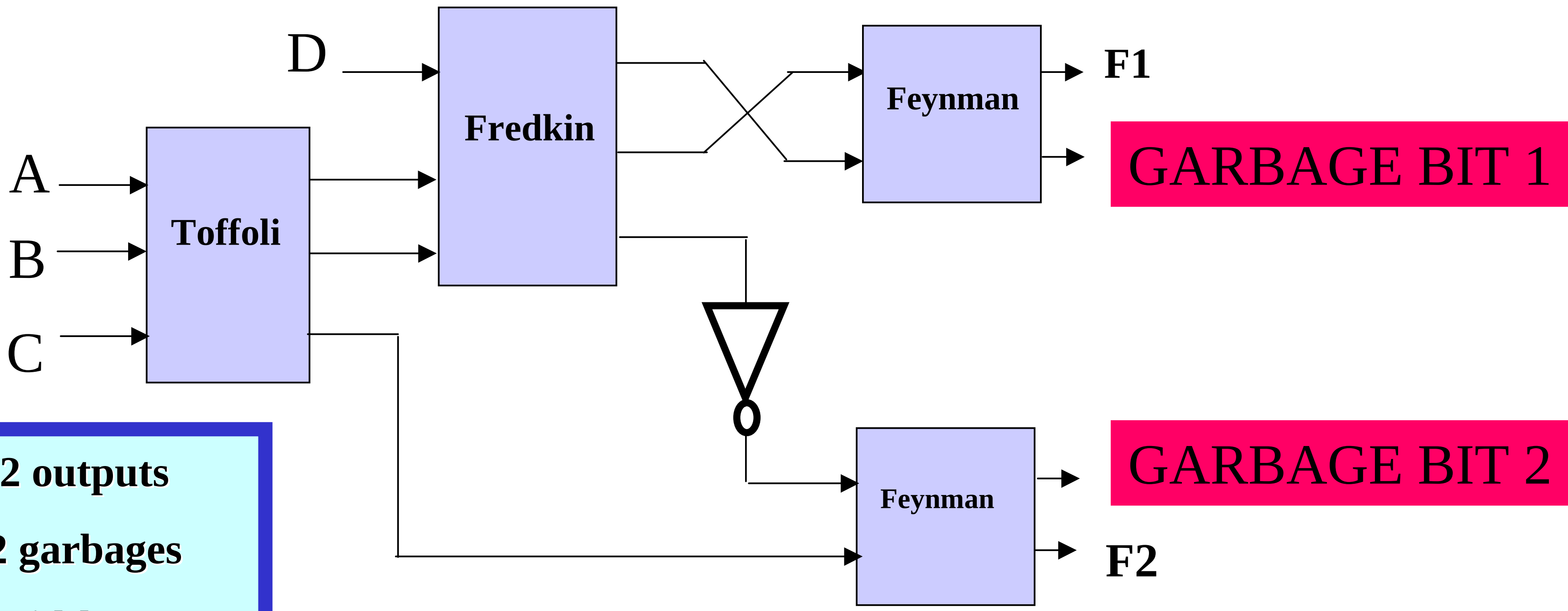
- When $C=1$ then $P = A + B$, $Q = A * B$, $R = \neg B$, so ***AND/OR*** gate is realized on outputs ***P*** and ***Q*** with ***C*** as the controlling input value.
- When $C = 0$ then $P = \neg A * \neg B$, $Q = A + \neg B$, $R = A \oplus B$.
- 18 different cofactors!

Kerntopf Gate

- As we see, the 3*3 Kerntopf gate is not a one-through nor a two-through gate.
- Despite theoretical advantages of Kerntopf gate over classical Fredkin and Toffoli gates, so far there are no published results on realization of this gate.

Logic Synthesis for Reversible Logic

How to build garbage-less circuits



2 outputs

2 garbages

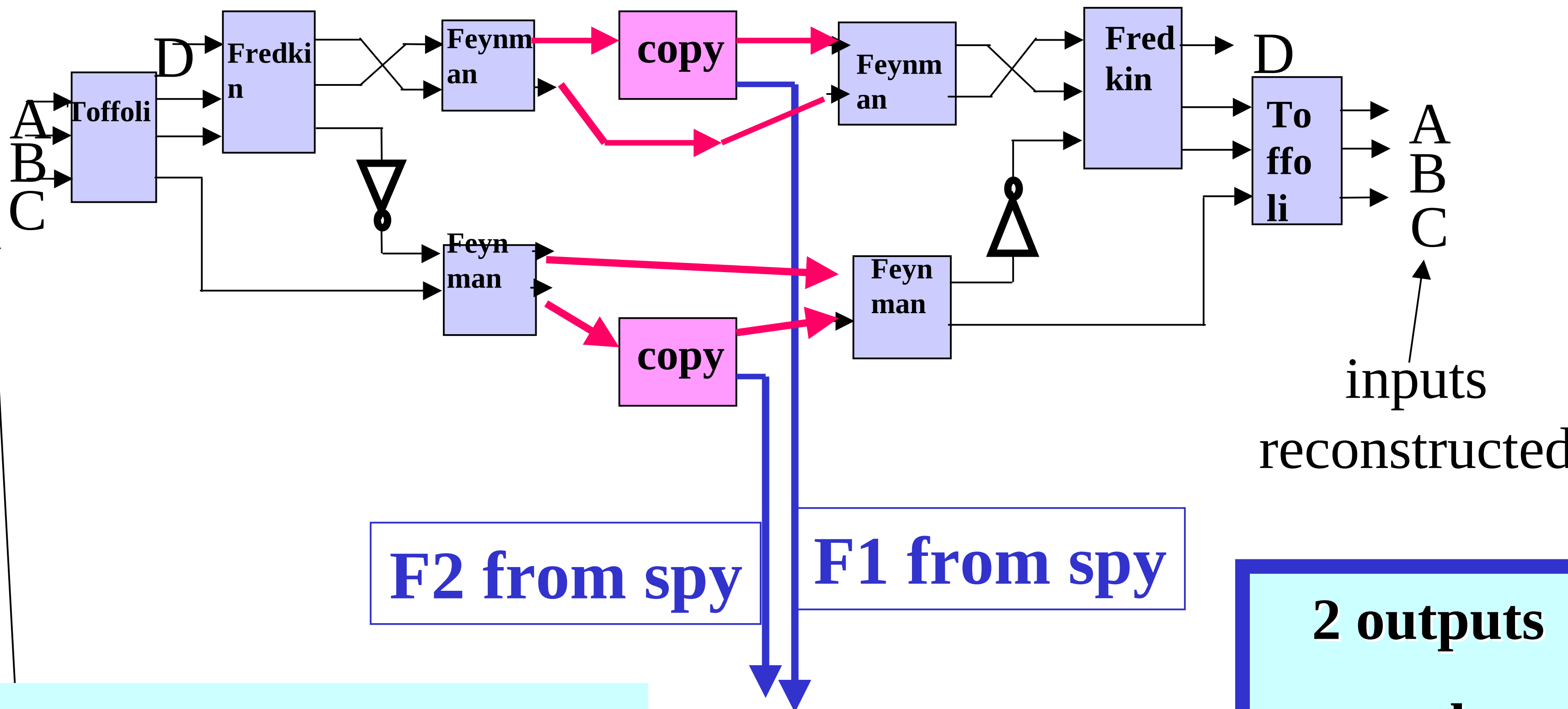
width = 4

delay = 4

We can decrease garbage at the cost of delay and number of gates

We create inverse circuit and add spies for all outputs

How to build garbage-less circuits



A,B,C,D are original inputs

This process is informationally reversible

It can be in addition thermodynamically reversible

2 outputs

no garbage

width = 4

delay = 9

Observations

- We reduced garbage at the cost of delay and number of gates
- We were not able to reduce the width of the scratchpad register

Goals of reversible logic synthesis

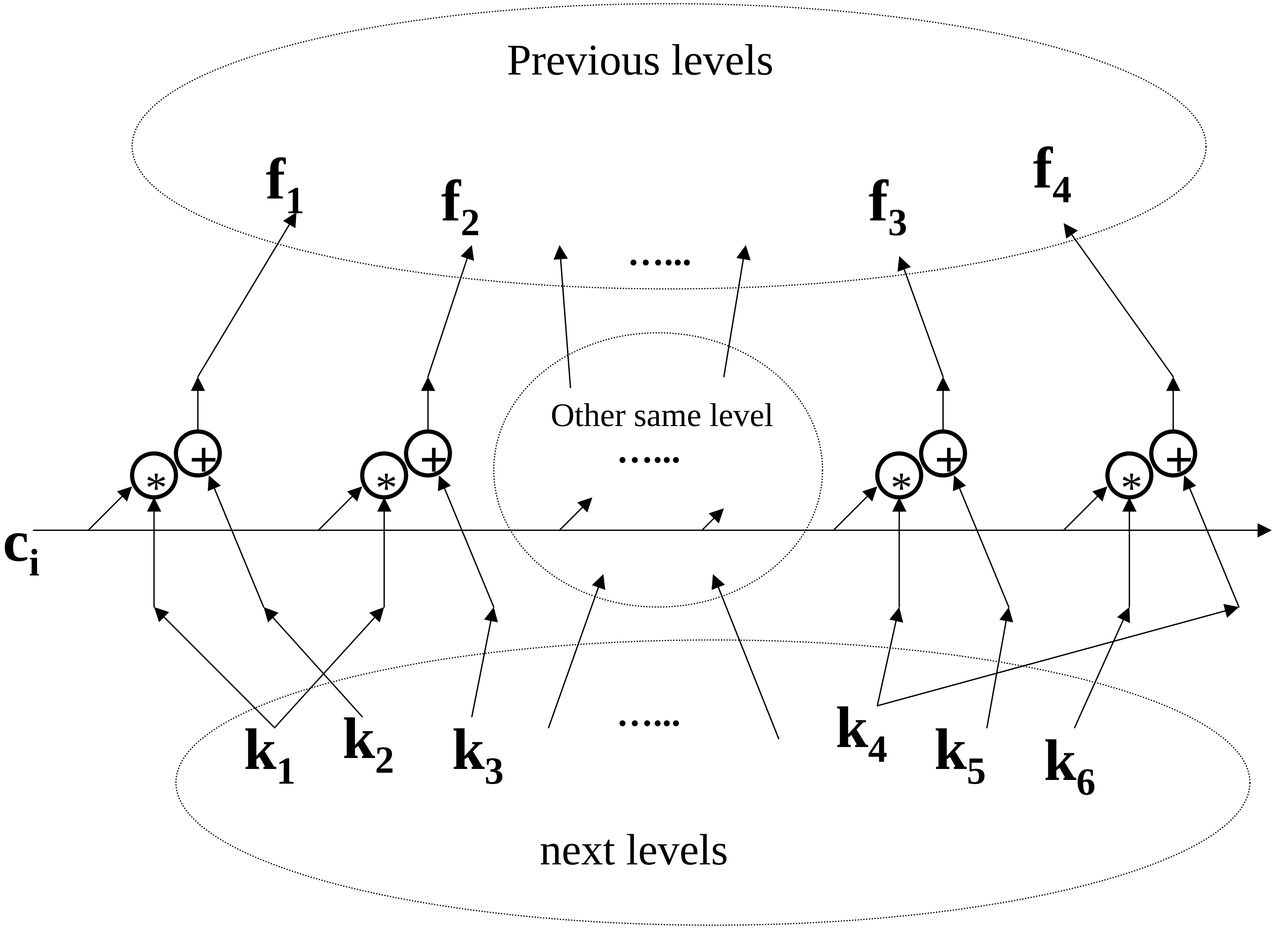
1. Minimize the garbage
2. Minimize the width of scratchpad register
3. Minimize the total number of gates
4. Minimize the delay

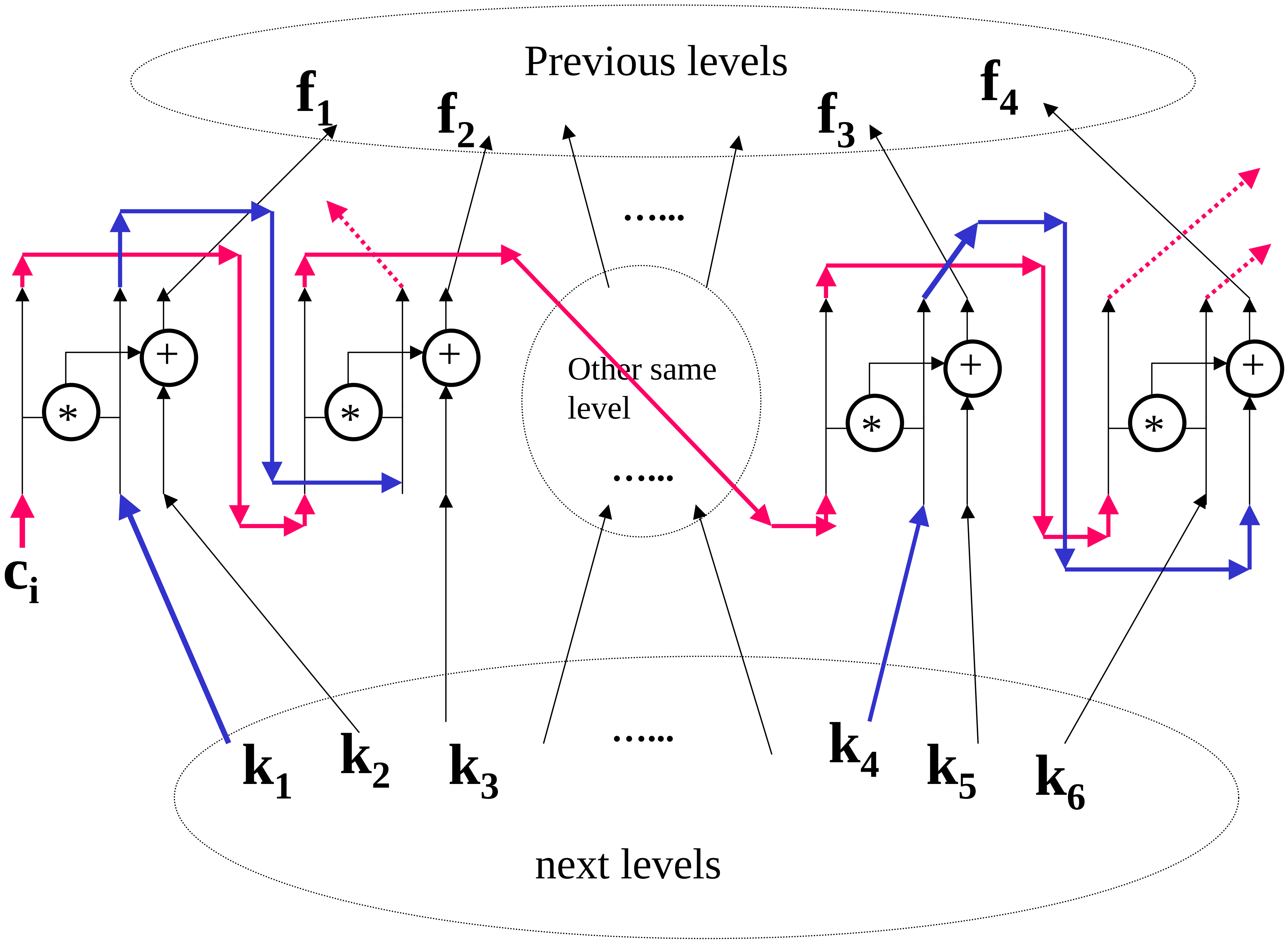
Multi-purpose Portland Decomposition

To distinguish this new general decomposition from the well-known decompositions of Ashenhurst, Curtis or Shannon, we call it the ***Multi-purpose Portland Decomposition***, the ***MP-decomposition*** for short.

Preprocessing

Synthesis from (p)KFDDs

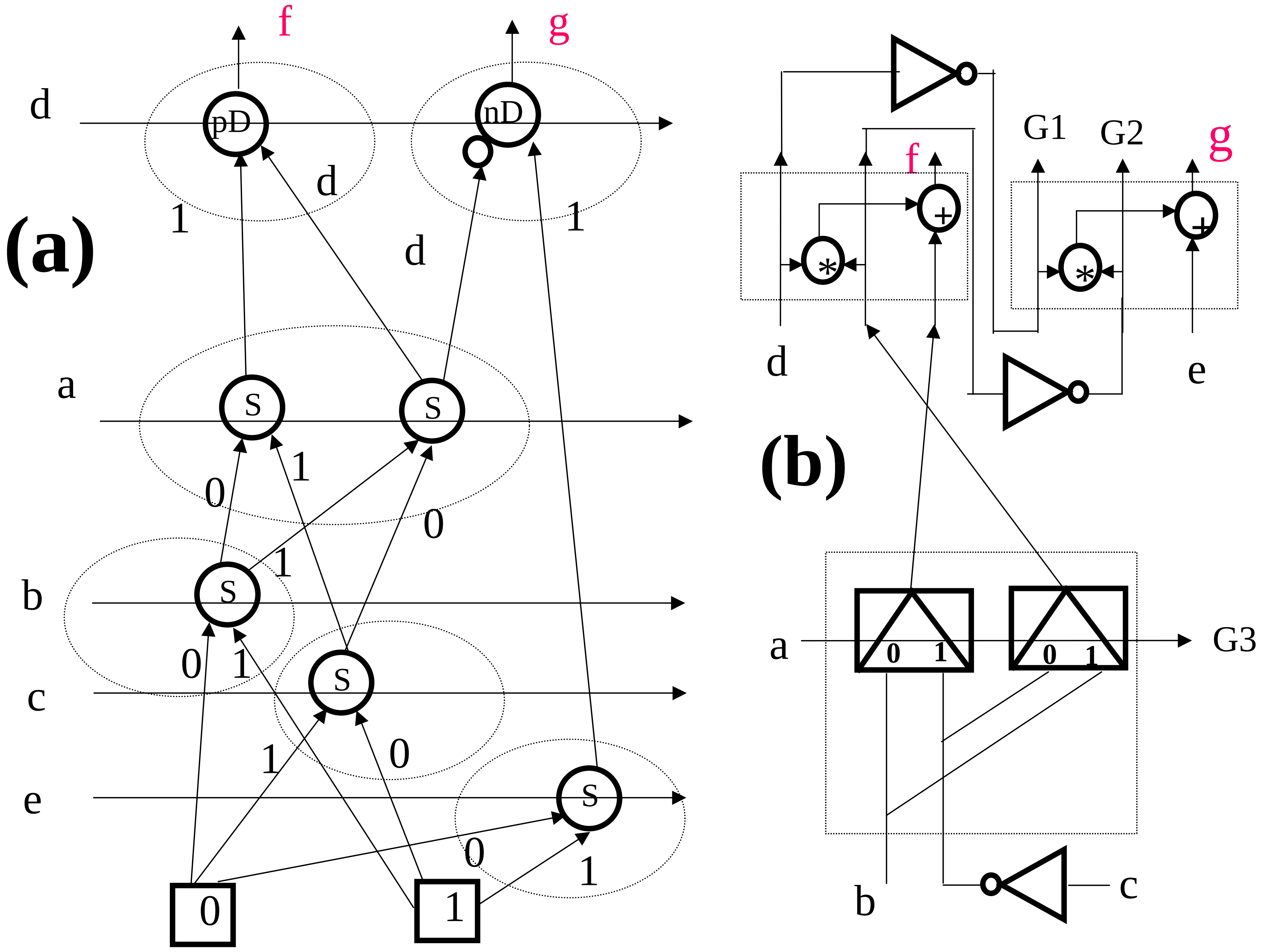

Previous levels
f1
f2
f3
f4
……
Other same level
……
ci
k1
k2
k3
……
k4
k5
k6
next levels



Previous levels
f_1
f_2
f_3
f_4
……
Other same level
……
c_i
k_1
k_2
k_3
……
k_4
k_5
k_6
next levels

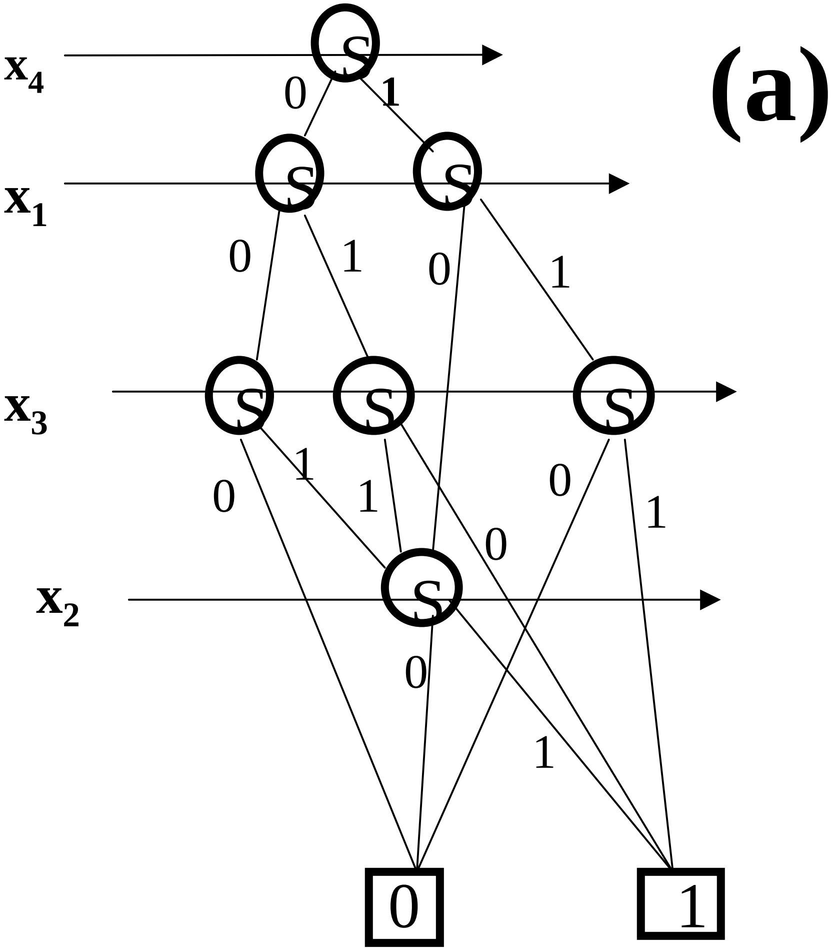
Example of converting a decision diagram to reversible circuit

Starting from Pseudo-Kronecker Functional Decision Diagram



f
g
d
pD
nD
d
1
d
1
(a)
a
S
S
1
0
0
1
b
S
0 1
c
S
1
0
e
S
0
1
0
1
G1
G2
g
f
+
+
*
*
d
e
(b)
a
0 1
0 1
G3
b
c

Starting from function-driven Decision Diagram



Standard BDD

(b)

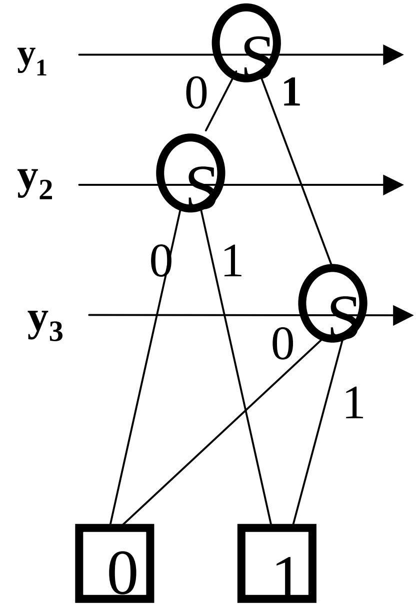
$y1 = x2 \oplus x3 \oplus x1x3$

$y2 = x3 \oplus x1 \oplus x1x4$

$y3 = x1 \oplus x4 \oplus x3x4$

$y4 = x4$

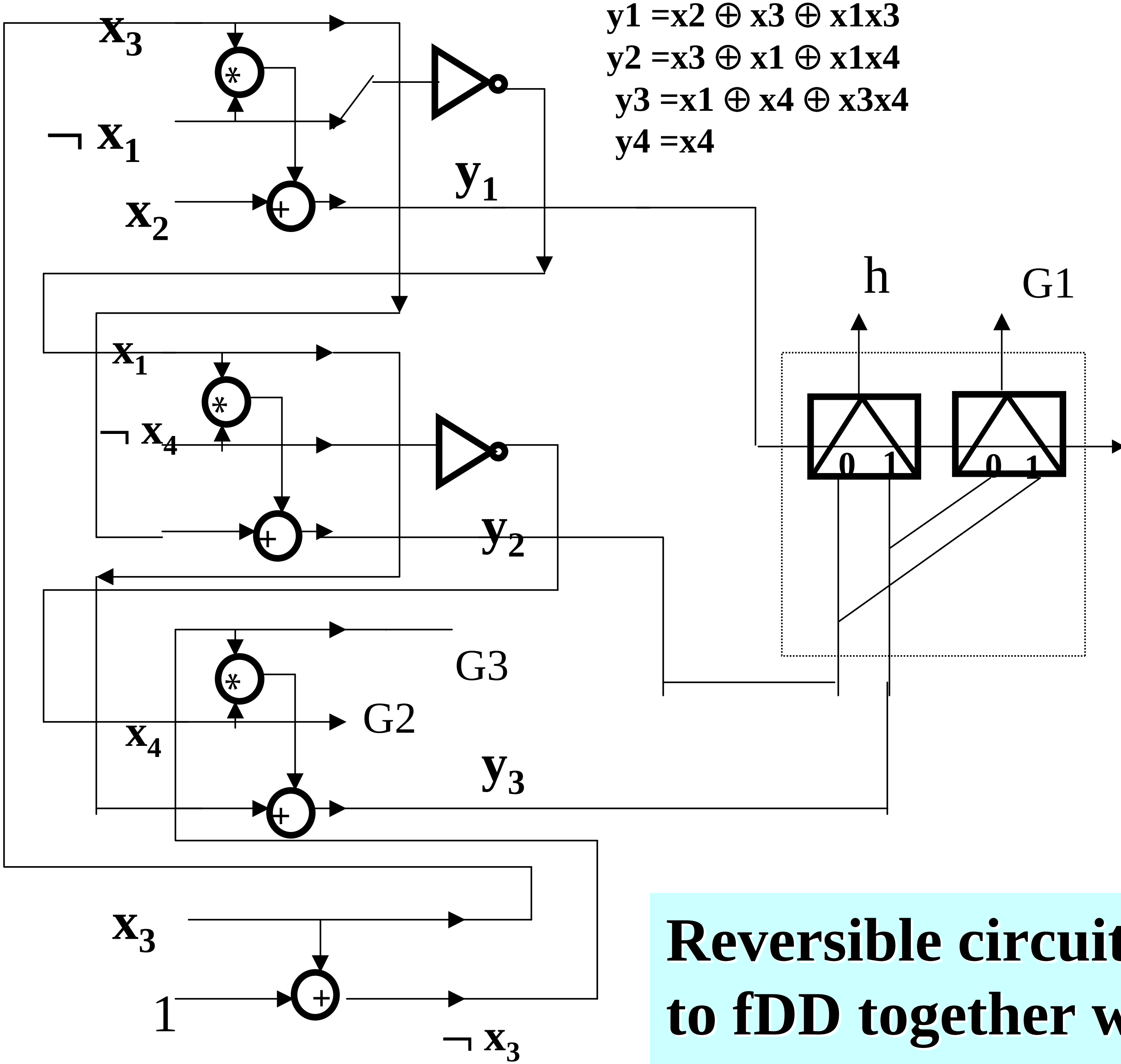
Nonlinear preprocessor



Corresponding fDD of Kerntopf

Converting fDDs to reversible circuits

$y1 = x2 \oplus x3 \oplus x1x3$
$y2 = x3 \oplus x1 \oplus x1x4$
$y3 = x1 \oplus x4 \oplus x3x4$
$y4 = x4$



Reversible circuit corresponding to fDD together with preprocessor

Open Problems:

- Is our set of mapping rules sufficient?
  - (we have currently about 20 rules, but many more can be created).
- What is the practically best starting point for large functions? KFDD? PKFDD? fDD?
- How to transform the diagram or the circuit to improve the cost function?
- What to do in case of rule conflict?
- How to create rules to decrease garbage?

Use of Reversibility

Observation:

Every synthesis method can be executed forward and backwards

Sometimes solution backwards can be simpler

Function F

A B C D	X Y Z V
0 0 0 0	***0 0 1 1***
0 0 0 1	***1 0 1 1***
0 0 1 0	***0 0 1 0***
0 0 1 1	***1 0 1 0***
0 1 0 0	0 0 0 0
0 1 0 1	0 1 1 1
0 1 1 0	0 0 0 1
0 1 1 1	0 1 1 0
1 0 0 0	*1 1 1 1*
1 0 0 1	*1 0 0 0*
1 0 1 0	*1 1 1 0*
1 0 1 1	*1 0 0 1*
1 1 0 0	1 1 0 1
1 1 0 1	0 1 0 1
1 1 1 0	1 1 0 0
1 1 1 1	0 1 0 0

Function **F** $^{-1}$

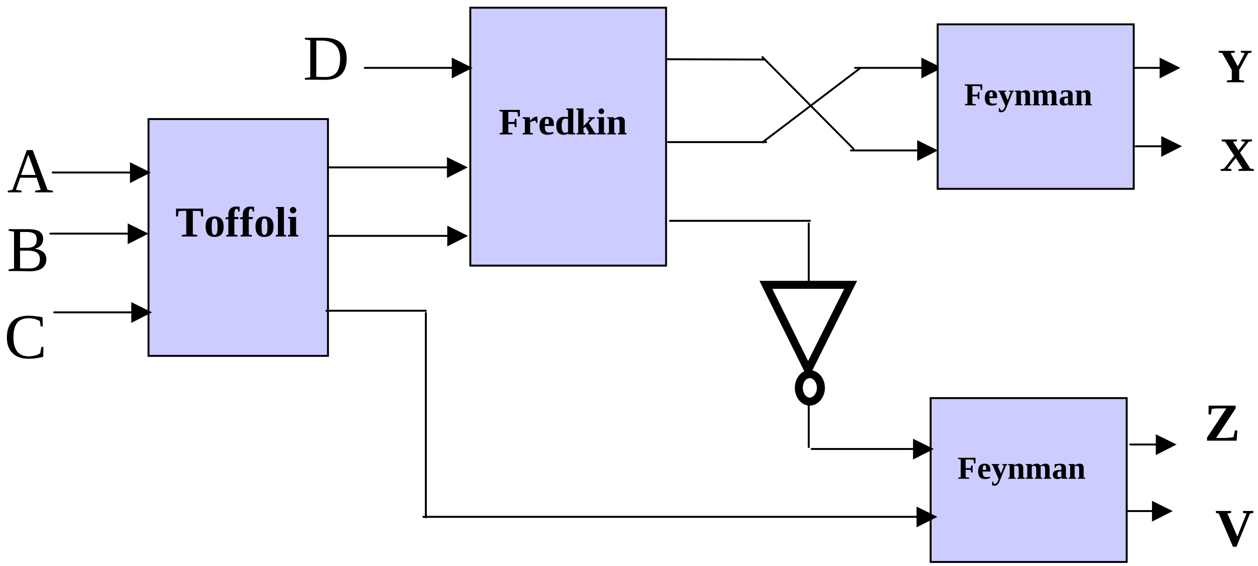
X Y Z V	A B C D
0 0 0 0	***0 1 0 0***
0 0 0 1	***0 1 1 0***
0 0 1 0	***0 0 1 0***
0 0 1 1	***0 0 0 0***
0 1 0 0	1 1 1 1
0 1 0 1	1 1 0 1
0 1 1 0	0 1 1 1
0 1 1 1	0 1 0 1
1 0 0 0	*1 0 0 1*
1 0 0 1	*1 0 1 1*
1 0 1 0	*0 0 1 1*
1 0 1 1	*0 0 0 1*
1 1 0 0	1 1 1 0
1 1 0 1	1 1 0 0
1 1 1 0	1 0 1 0
1 1 1 1	1 0 0 0

Several methods exist to calculate inverse from forward function

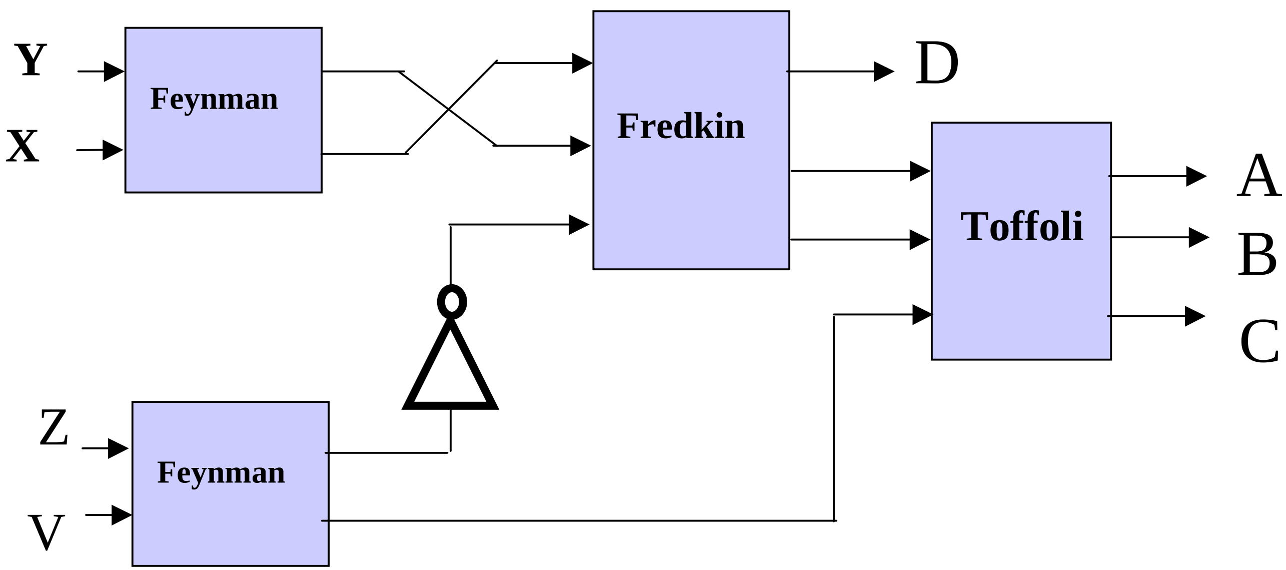
Forward Function

Inverse Function

Circuit for function F



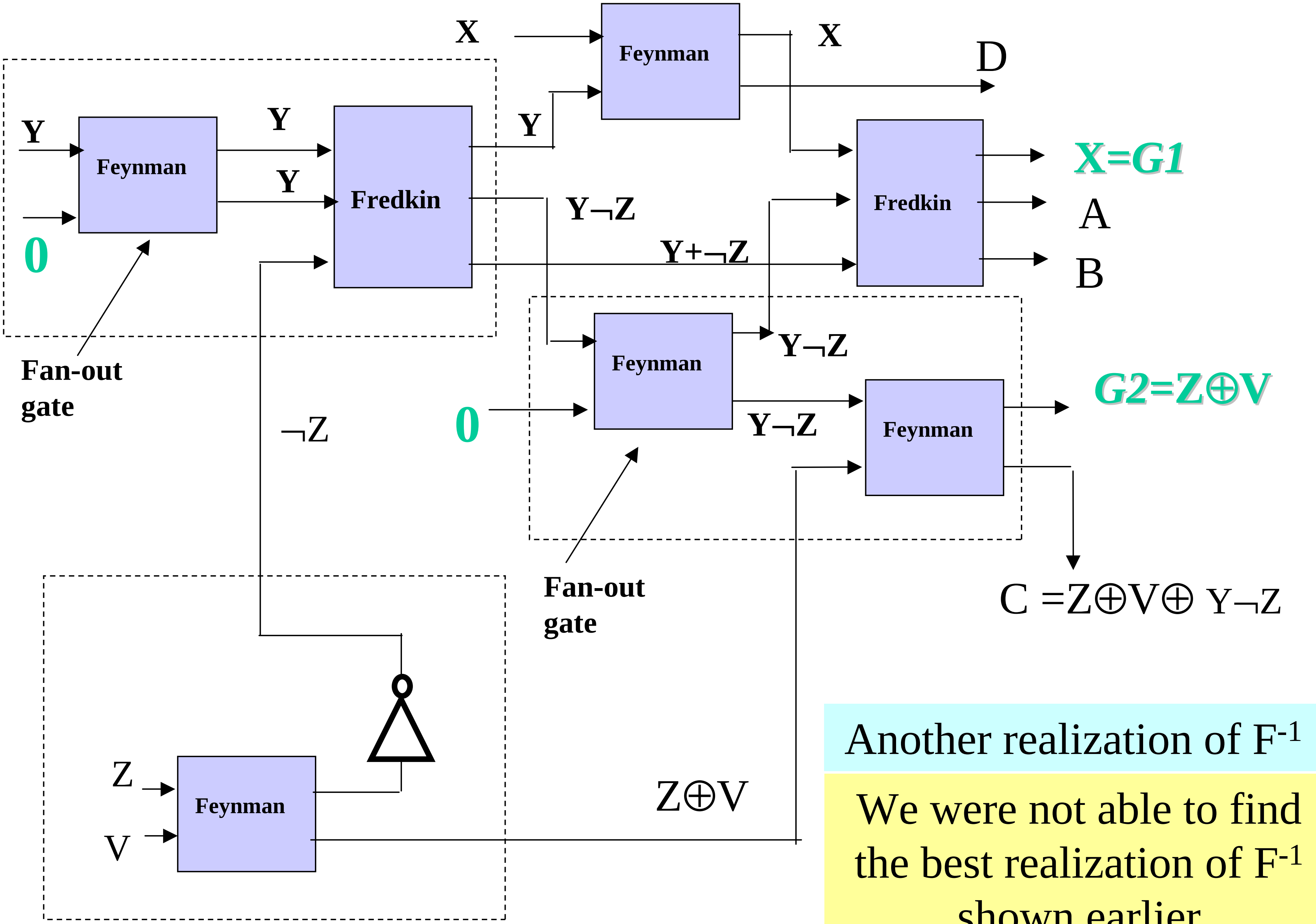
$$\boxed{F*F^{-1}=I}$$



Circuit for function F^{-1}

Method of realization of F

- **1.** Find function inverse **F^{-1}** of function F.
- **2.** Synthesize **F^{-1}** using any method for reversible gates
- **3.** Draw the schematics of **F^{-1}** from gates.
- **4.** In the schematics replace every gate by its reverse and change the direction of signals.
- **5.** The new schematics is the realization of **F**.



Another realization of F^{-1}

We were not able to find the best realization of F^{-1} shown earlier

Compositions and Decompositions

Composition Methods

- Composition methods with matching-based cell selection and complexity measures have been presented by:
  - Dietmeyer
  - Schneider
  - Wojcik
  - Michalski
  - Jozwiak, Chojnacki and Volf
  - DeMicheli library matching
  - Kravets and Sakallah

Uses library of 1*1,2*2 and 3*3 cells

For every 3*3 function, a ready solution cascade is stored - see the results of Kerntopf and Storme-DeVos

For every reversible gate, all cofactors can be stored (constants and garbage) and used in NPN matching.

Gates with more cofactors (like Kerntopf) are better

Uses equivalence mapping transformations based on NPN-equivalence

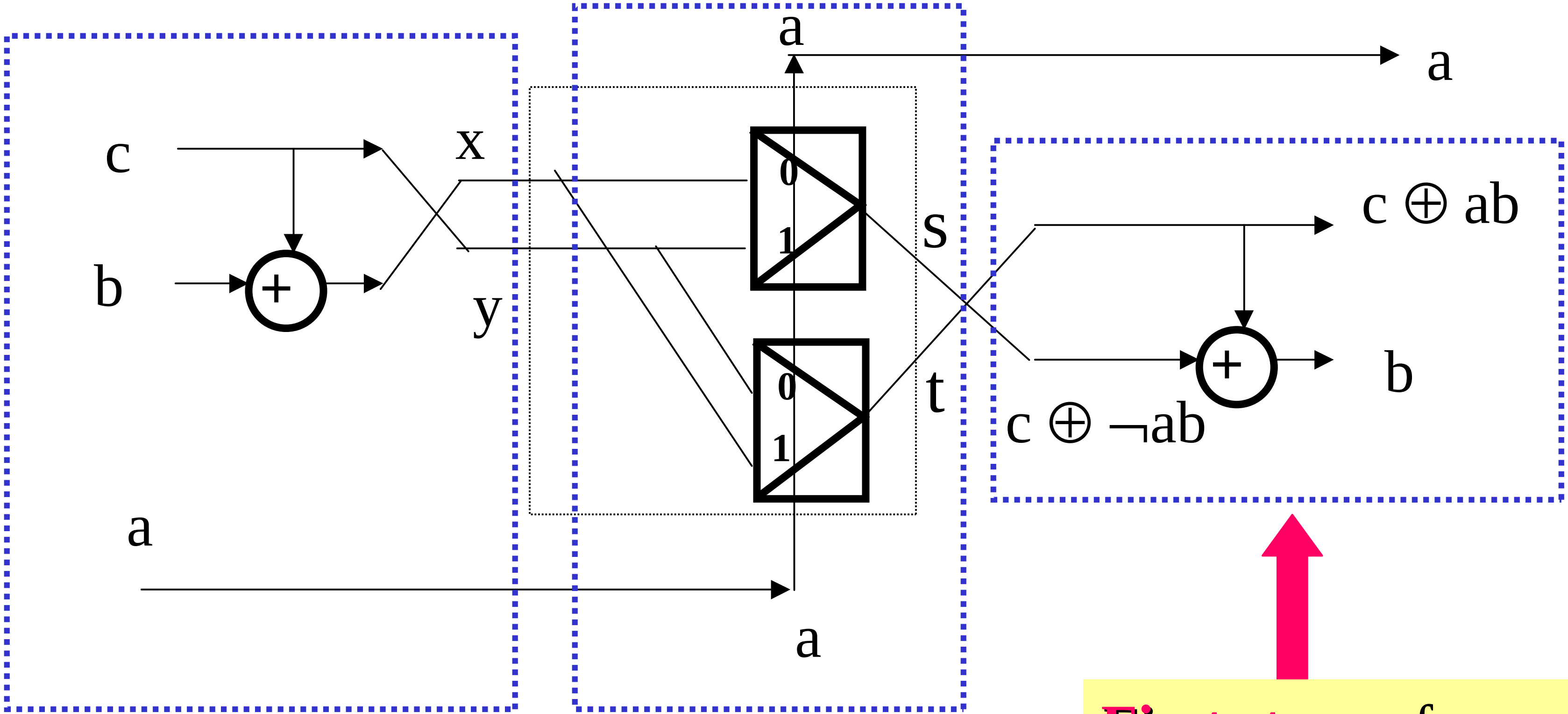
Synthesis from inputs to outputs and from outputs to inputs, backtracking and look-ahead strategies

Can be applied to both classical reversible and quantum logic

All intermediate functions calculated in terms of input variables

How to select the cell, its alignment and constants?

- Select the gate that provides smallest complexity evaluation of the remaining functions and intermediate functions.
- This works for both forward and backward transformations
- I propose PPRM for matching because it is easy to implement. Other representations and measures can be used for matching - De Micheli, Dietmeyer, Jozwiak.



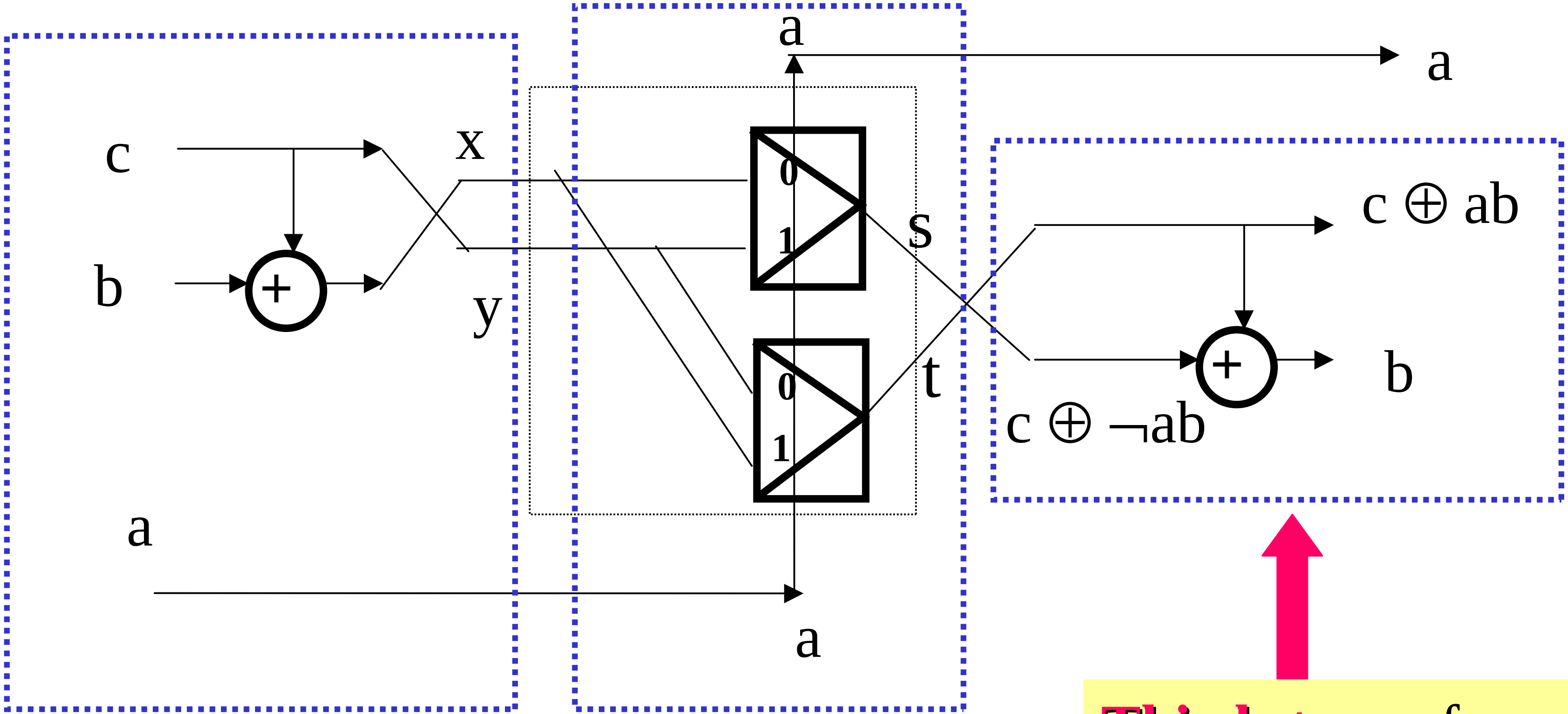
First stage of decomposition: Feynman gate

Third stage of decomposition: Feynman gate

Second stage of decomposition: Fredkin gate

Decompositional synthesis of Toffoli Gate from Fredkin and Feynman Gates

Using PPRM representation allows for NPN matching and good evaluation of remaining function complexity



Third stage of composition: Feynman gate

First stage of composition: Feynman gate

Second stage of composition: Reversible Expansion for Fredkin gate

NPN matching takes care of permutations and inverters

Compositional synthesis of Toffoli Gate from Fredkin and Feynman Gates

Composition

From inputs to outputs

What to do if the initial function is not reversible?

Composition

Restrict to C=0

Toffoli →

A B C	X Y Z	h=A⊕B	g=A*B	G
0 0 0	0 0 0	0	0	-
0 0 1	0 0 1	0	0	-
0 1 0	0 1 0	1	0	-
0 1 1	0 1 1	1	0	-
1 0 0	1 0 0	1	0	-
1 0 1	1 0 1	1	0	-
1 1 0	1 1 1	0	1	-
1 1 1	1 1 0	0	1	-

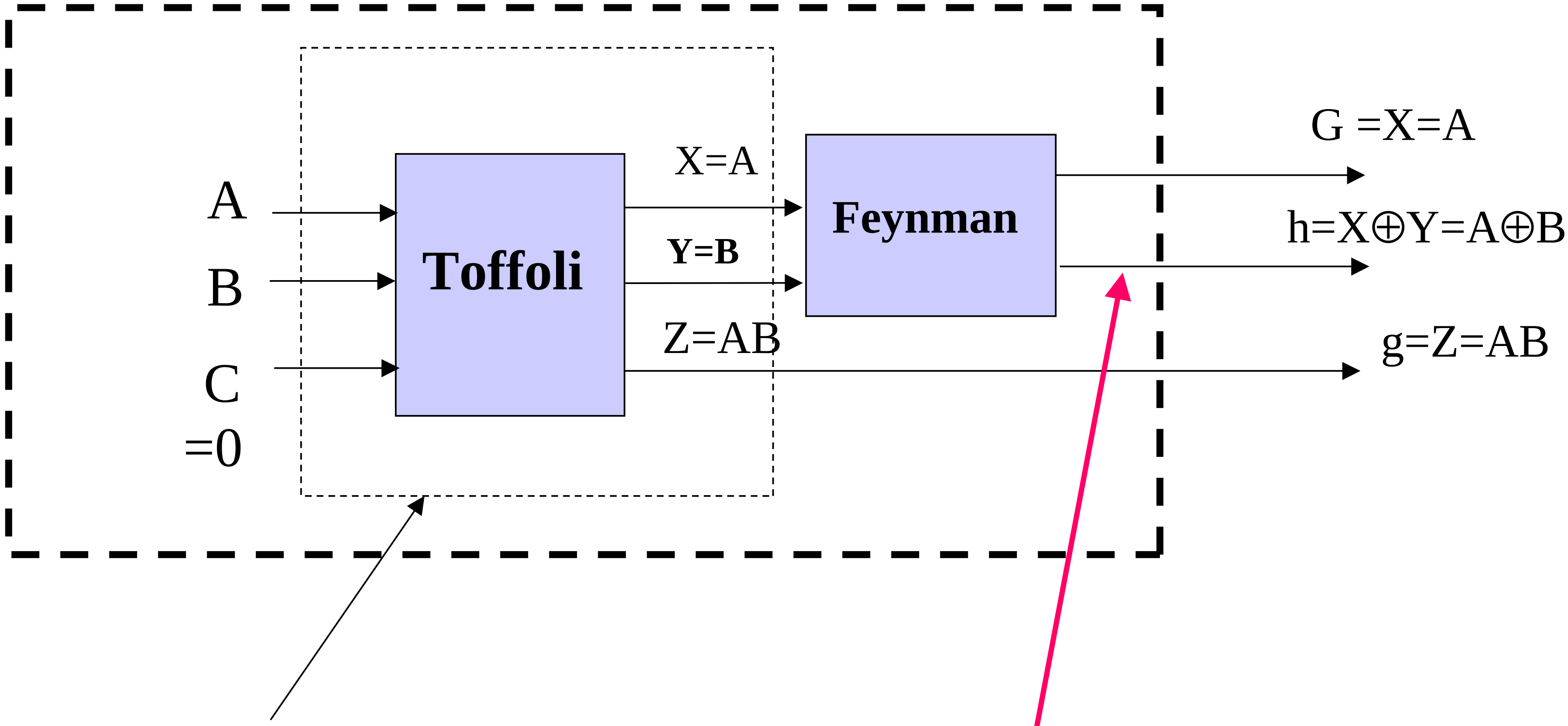
Z=A*B

1. Toffoli gate assumed.

2. New functions X,Y,Z created.

3. C=0 restriction taken

4. Functions h and g of X,Y,Z are created



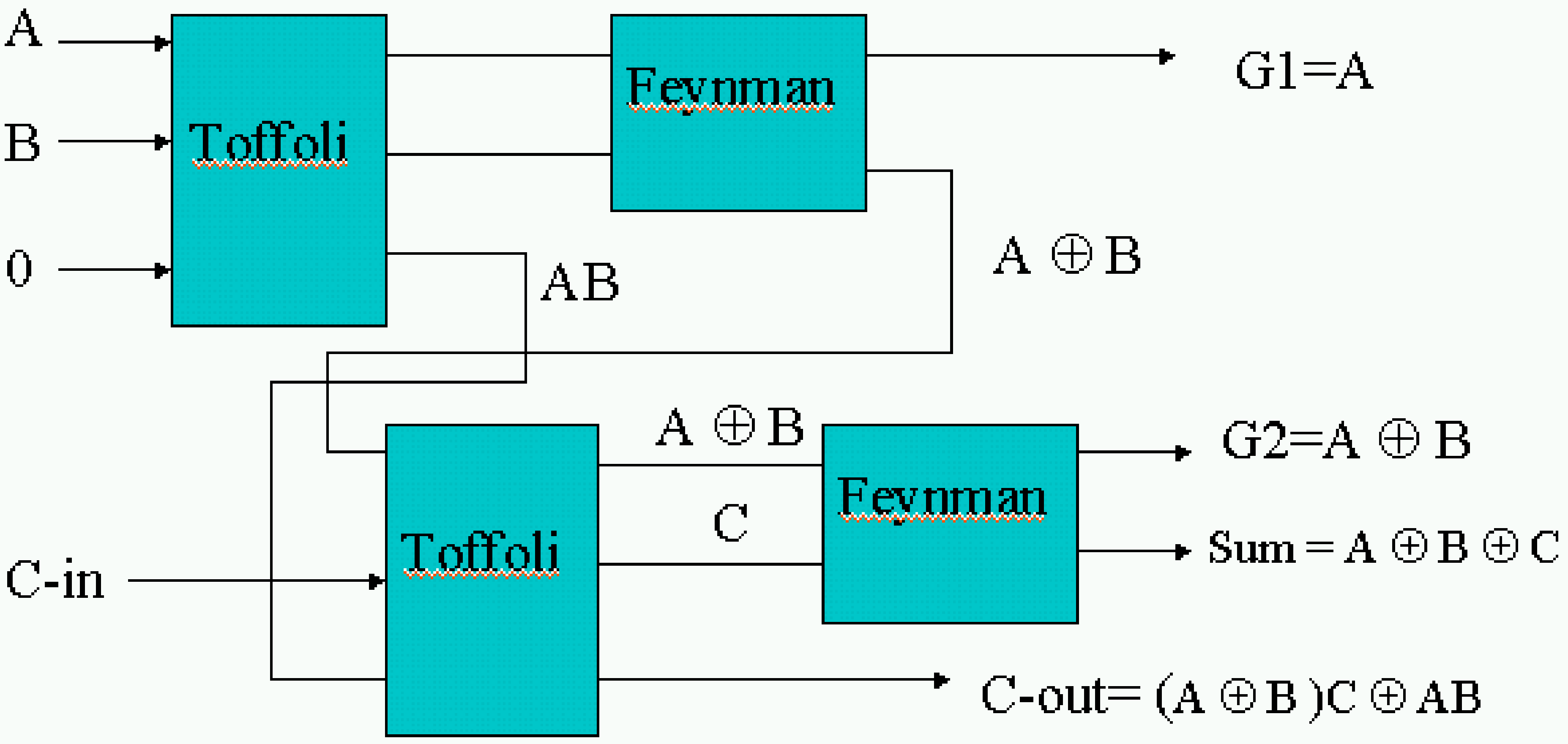
Created first

Calculated in terms of new variables X,Y,Z and selected Feynman gate

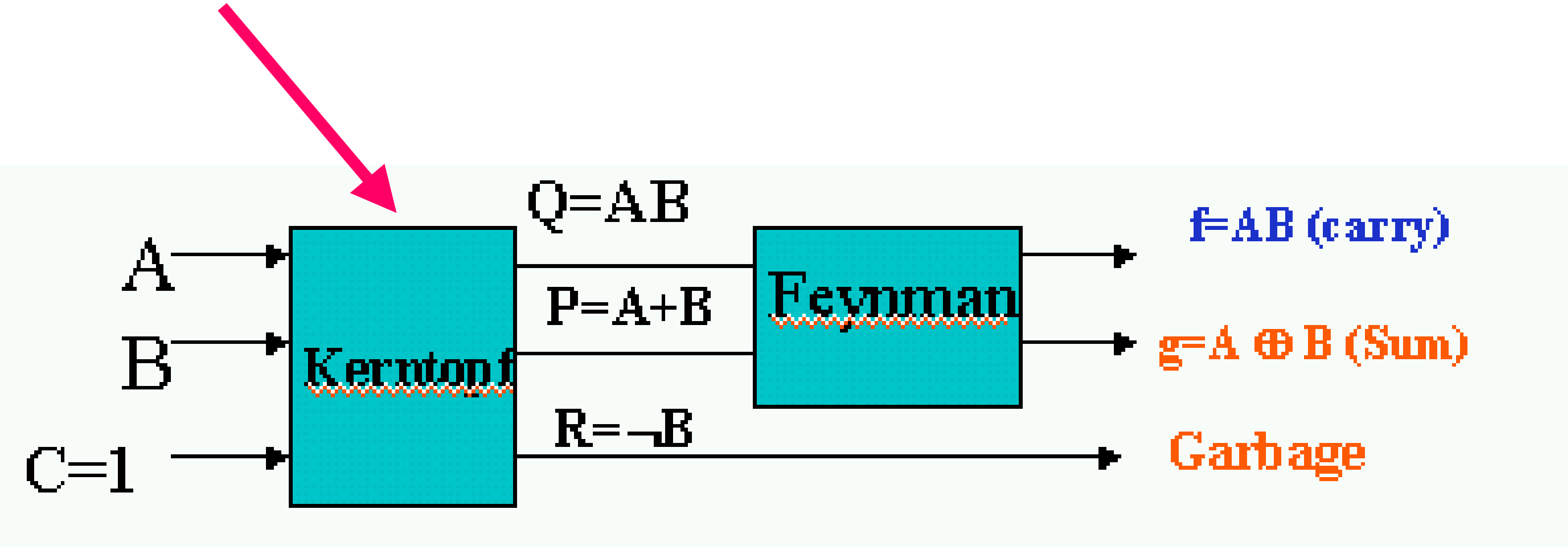
Compositional synthesis of non-reversible function of half-adder

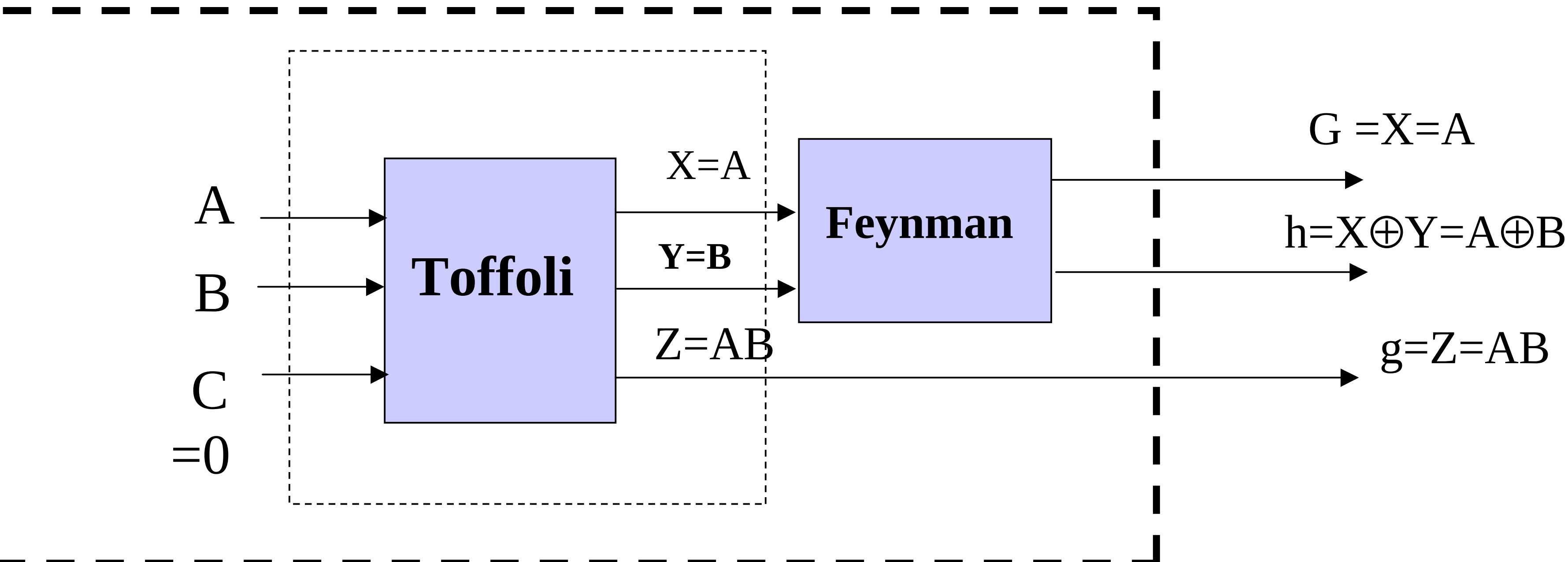
Adder is composed from half-adders

2 garbage bits and one constant

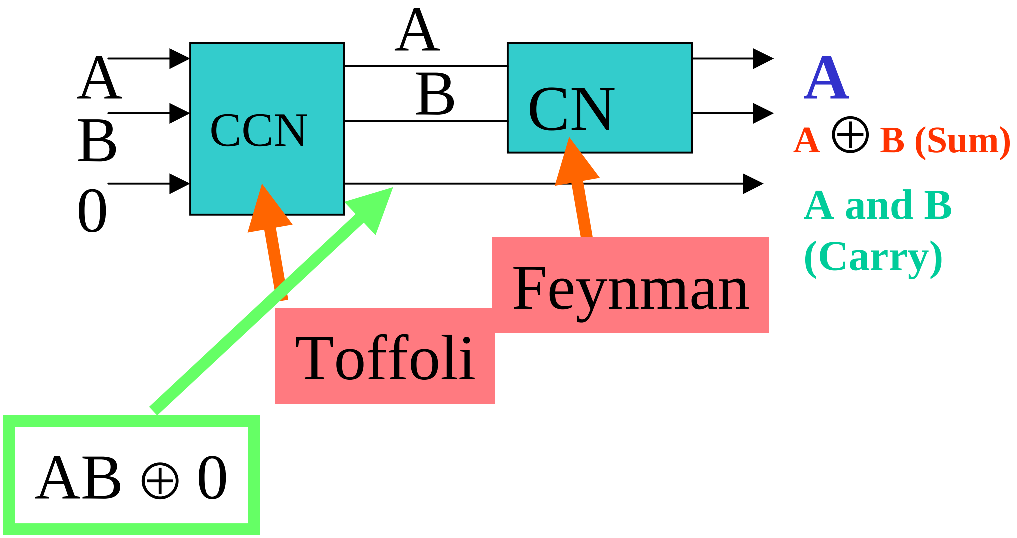


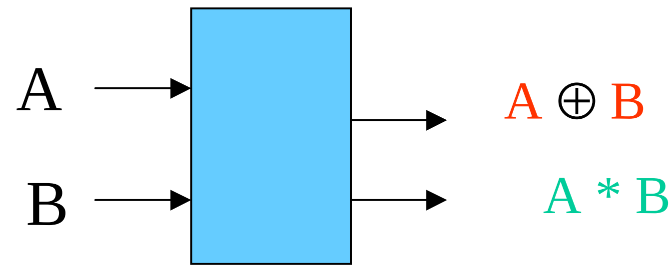
Another variant assumes Kerntopf gate





Heuristics for finding simplest reversible function during composition





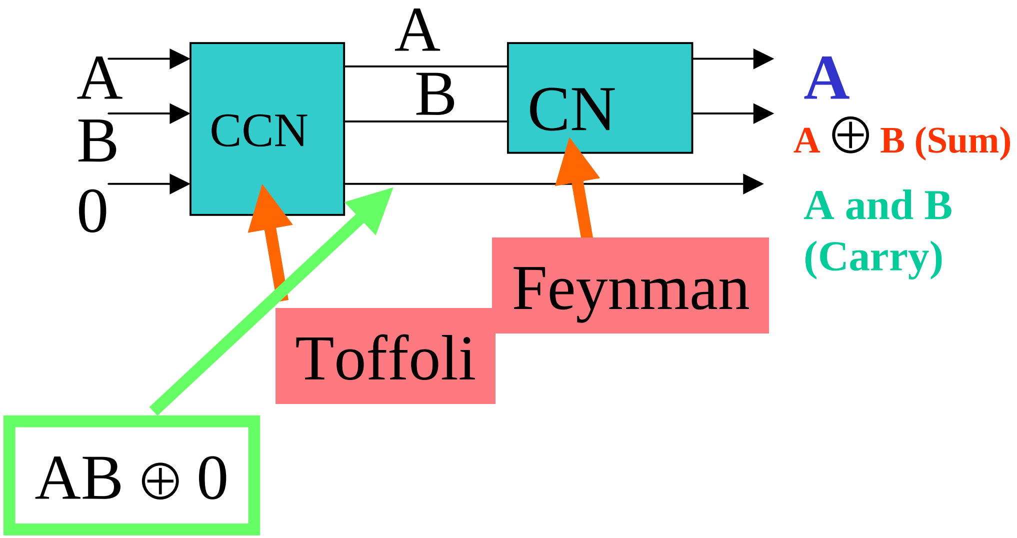
Mapping of a half-adder

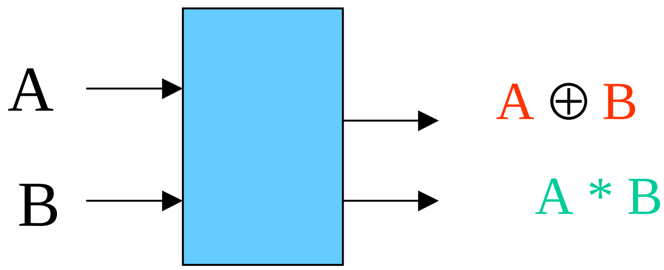
AB	00	01	11	10
	00	10	01	10

This is what we would like to have

But this is not reversible,

Heuristics for finding simplest reversible function during composition



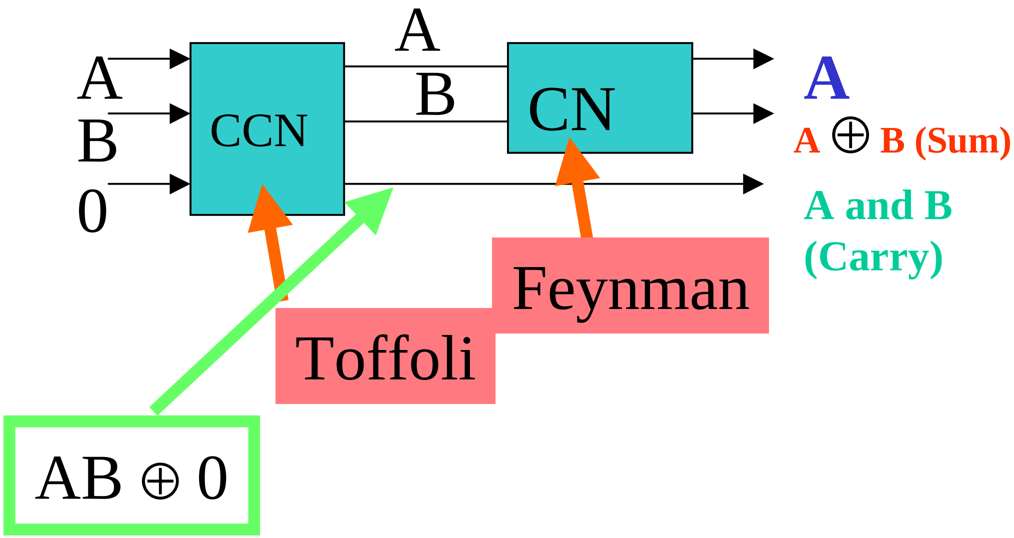


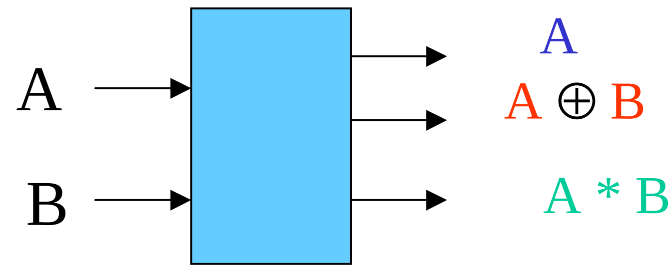
Mapping of a half-adder

AB	00	01	11	10
	00	10	01	10

The simplest way to separate them is to add variable A

Heuristics for finding simplest reversible function during composition





Mapping of a half-adder

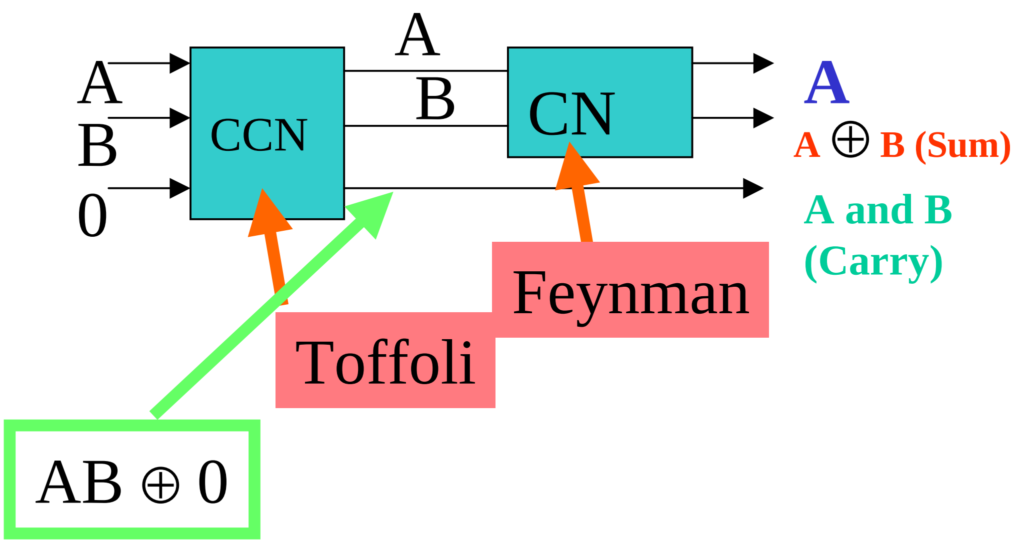
AB	00	01	11	10
	000	010	101	110

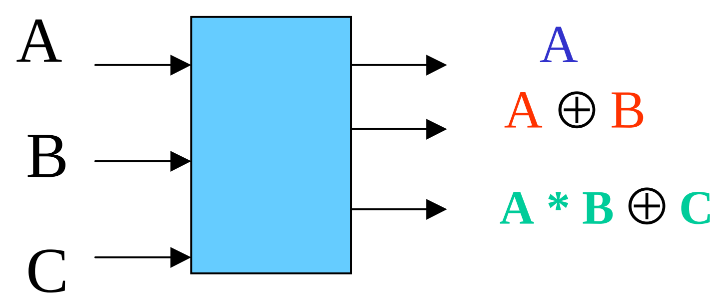
This is what we would like to have

But this is not reversible,

because more outputs than inputs

Heuristics for finding simplest reversible function during composition





Mapping of a half-adder

C \ AB	00	01	11	10
0	0: 000	1: 010	3: 101	2: 110
1	4: 001	5: 011	7: 100	6: 111

Positive Davio Gate

Now <u>function is reversible</u> of A,B,C arguments. But we still need to decompose it to known gates

Search

- Search is defined by:
  - selecting function to be realized
  - selecting a gate type
  - selecting its forward or backwards matching
  - selecting other signals to match
  - selecting constant

Garbage functions are becoming less and less undefined in the process of decomposition

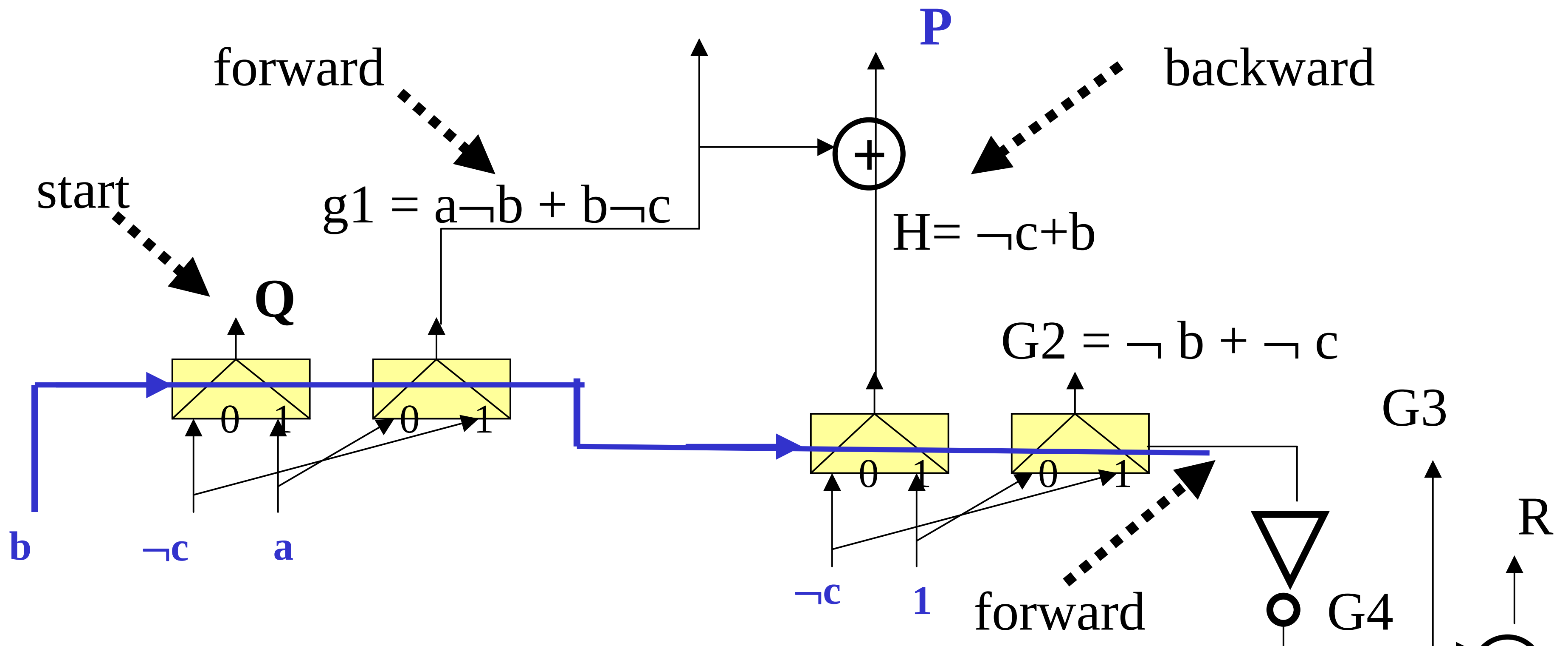
They can be used for synthesis

We adopt classical AI search algorithms

(depth-first, breadth-first, A*,best bound, etc)

Search

- Search cannot be avoided
- Trade-off between quality of solution and search time
- The only real improvement is possible through better selection heuristics and better implementation of cell library.



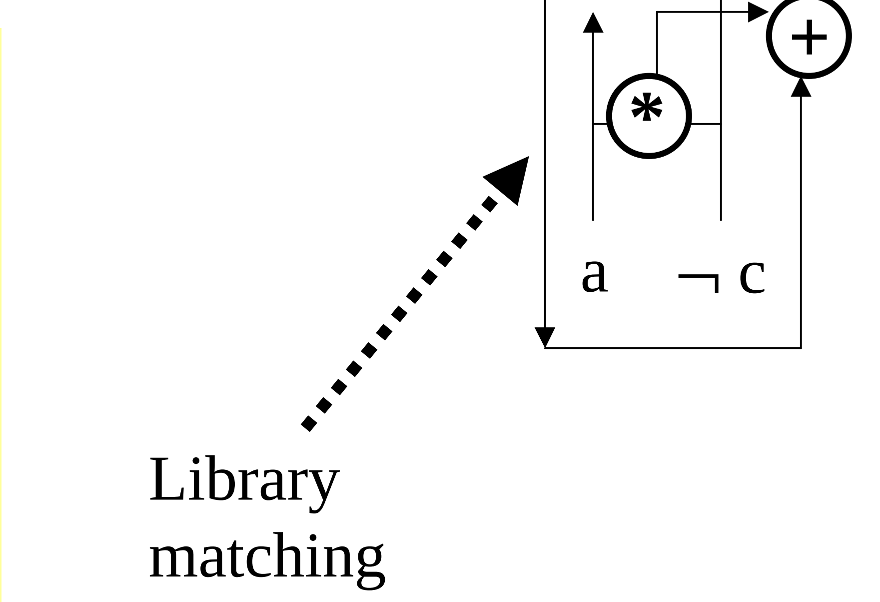
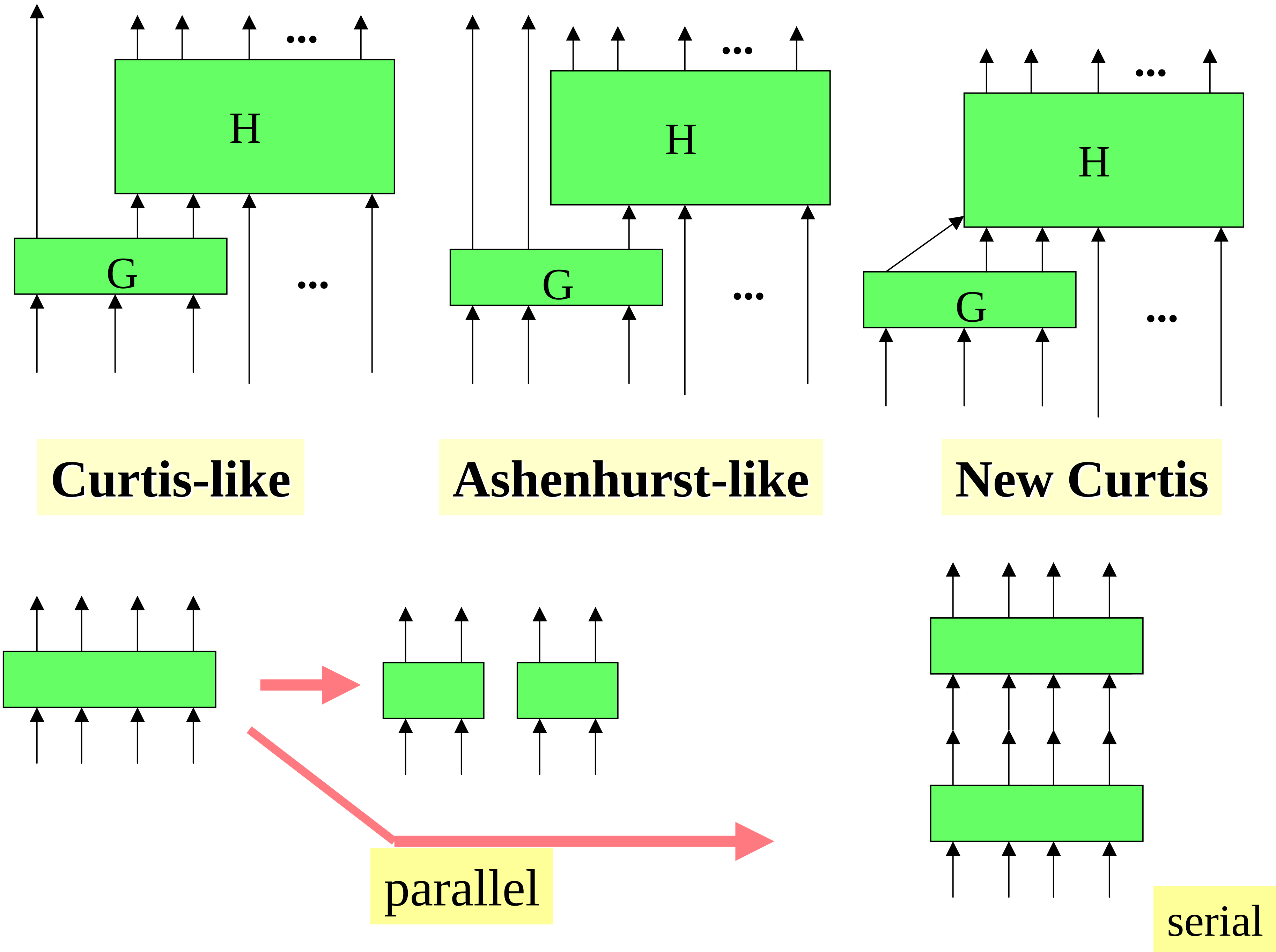


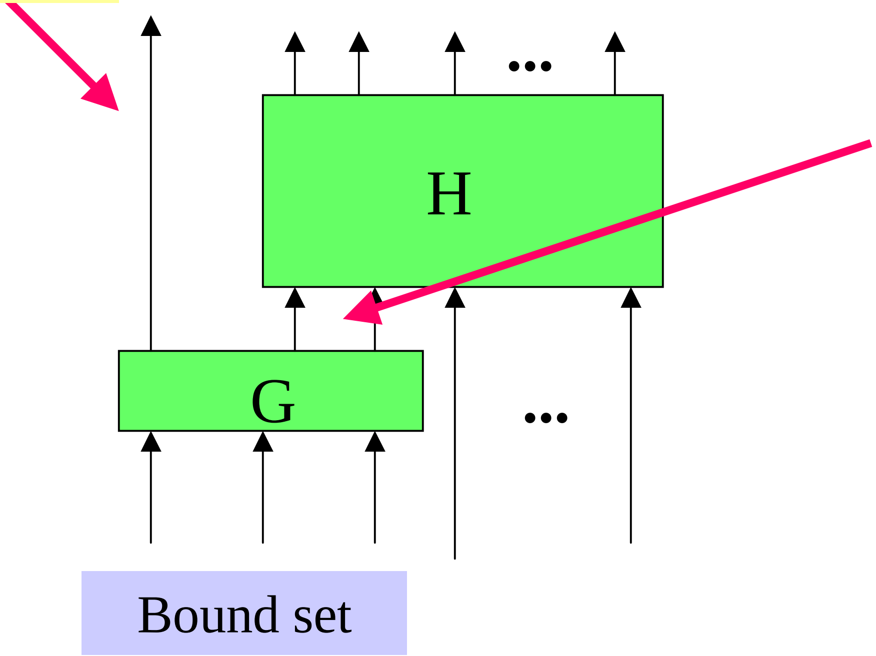
Illustration of ***cooperation of various methods*** *to design the Kerntopf gate from Fredkin, Toffoli, Feynman gates and inverters. One fan-out gate for input a, inverter for c and two fan-out gates for ¬c are not shown.*

Adaptations of functional decomposition methods


H
G
H
G
H
G
Curtis-like
Ashenhurst-like
New Curtis
parallel
serial

Curtis Decomposition

We sacrifice this wire for garbage



These two signals should be encoded using 4 symbols

- During graph coloring of an **incompatibility graph** for nodes with minterms of bound set, two conditions are satisfied:
  - there must be 4 colors

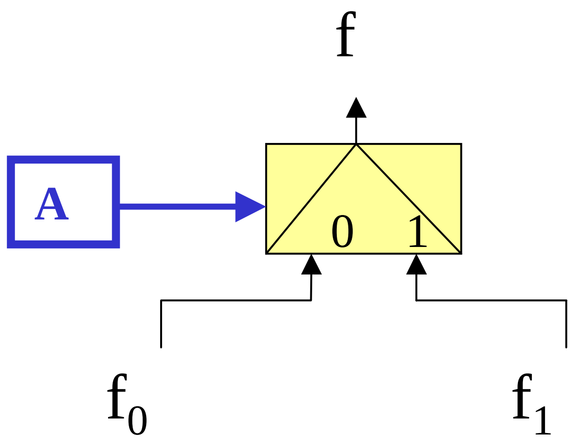
Curtis Decomposition

- Probability of finding such coloring is increased by having more don’t cares
- More don’t cares are created by repeating variables in bound and free sets.
- This principle is the base of all decompositions of reversible functions and relations.

Levelized Structures

Two-Dimensional Lattice Diagrams for reversible logic

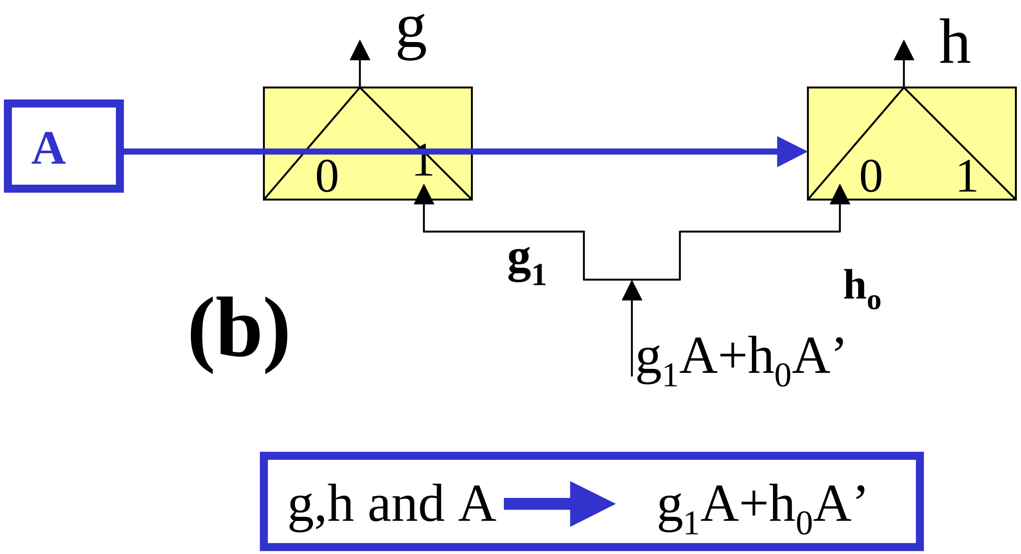
Three Types of General Expansions



Forward Shannon

f and A → f_0 and f_1

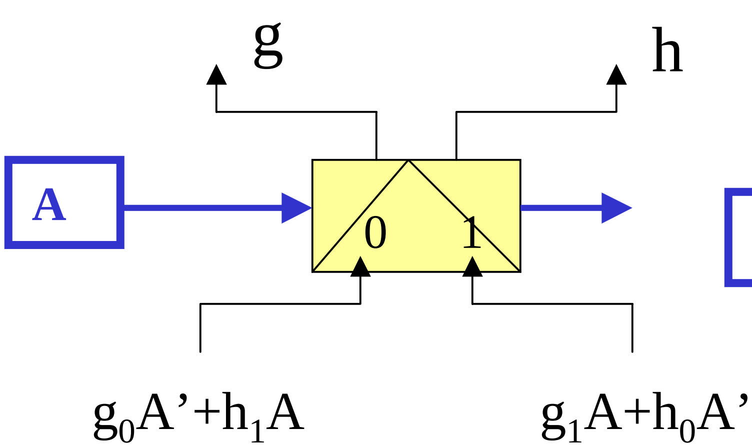
Three Types of General Expansions



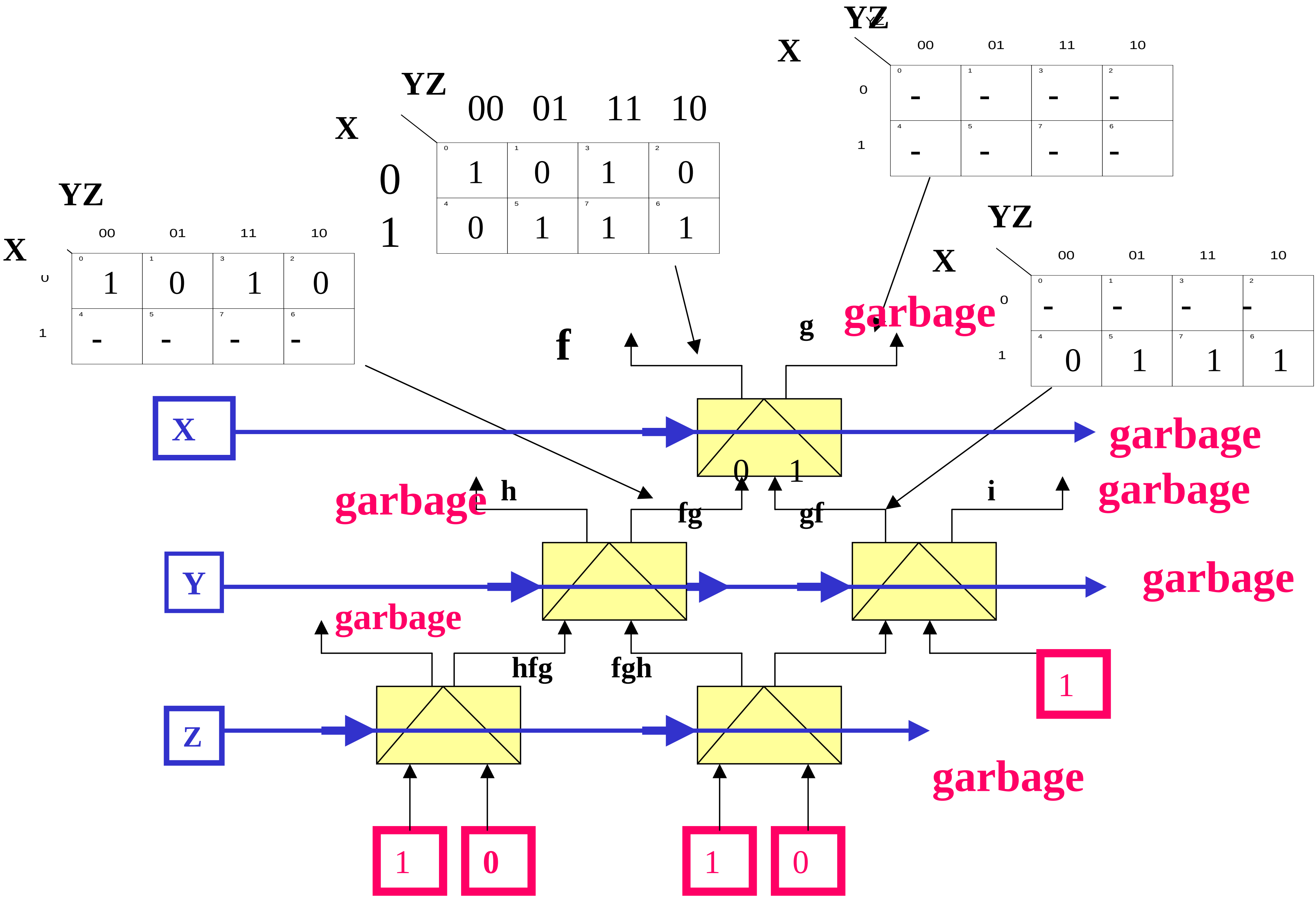
Reverse Shannon

Three Types of General Expansions

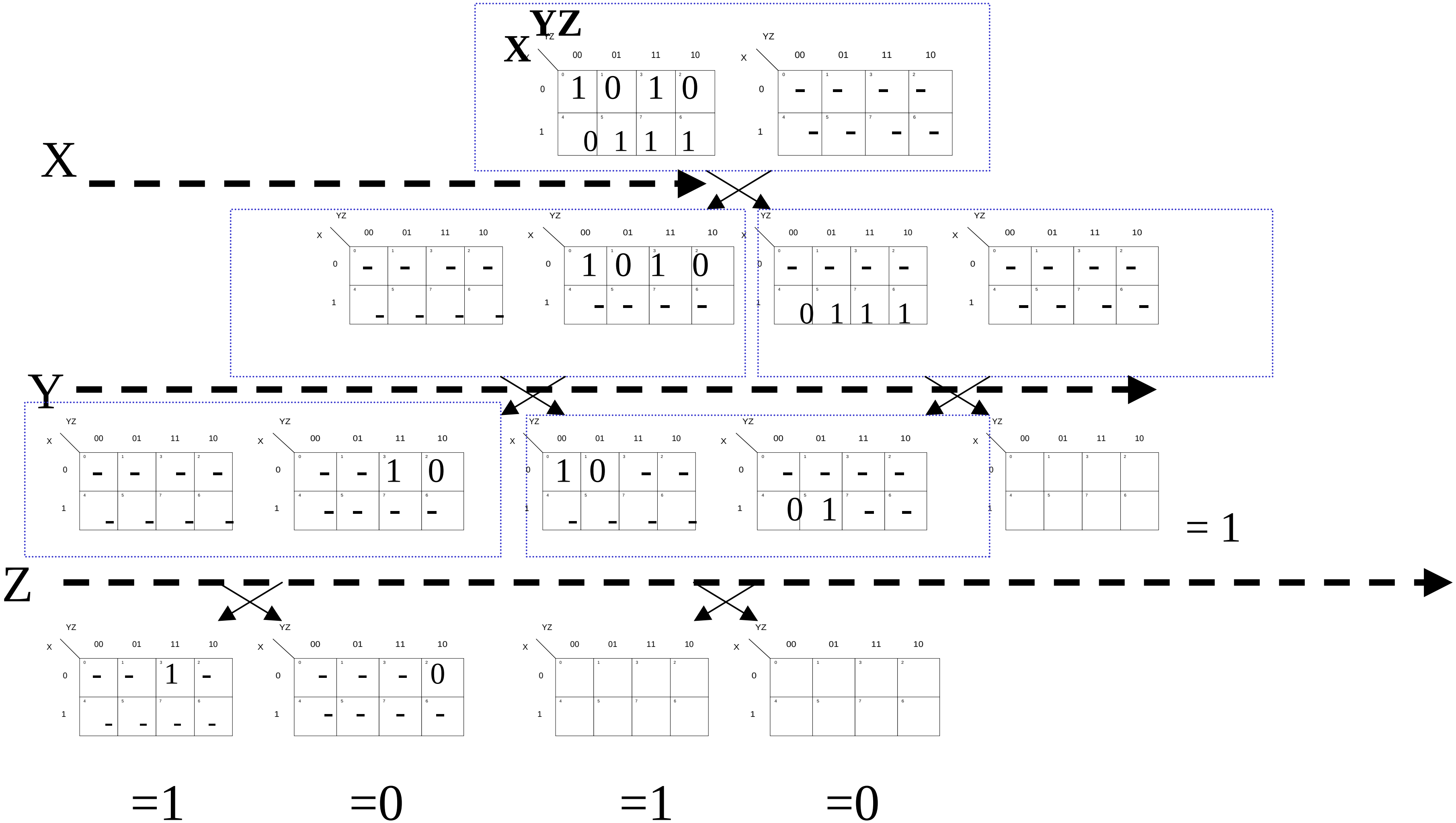
Reversible Shannon



$\boxed{g,\ h,\ \text{and}\ A \rightarrow g_0A'+h_1A\ \text{and}\ g_1A+h_0A'}$



YZ
X
00
01
11
10
0
1
-
YZ
X
00 01 11 10
0
1
1 0 1 0
0 1 1 1
YZ
X
00 01 11 10
0
1
1 0 1 0
- - - -
YZ
X
00 01 11 10
0
1
- 1 1 1
0 1 1 1
f
g
garbage
X
Y
Z
h
i
fg
gf
hfg
fgh
0
1
1
0
1
0
1



This method is fully algorithmic

Variable order selection problem

Expansion type selection problem

Applications of levelized expansion method

- This method can be used to create arbitrary circuits, not only lattices
- Any constraint on layout size or shape can be imposed.
- The expansion method can be used as the last-resort approach when other methods cannot find solution, in order to reduce the number of variables
- It introduces garbage and constants, but this is unavoidable when functions are not balanced
- When realizing multi-output functions start from those that are balanced or closest to balanced.

Conclusions

- New concepts:
  - ***(1)*** Reversible Shannon Expansion for ***k*k*** binary Fredkin Gates ***(k>2) and generalizations -*** **reversible decision diagrams,**
  - ***(2)*** Reversible Fredkin Lattice structures for logic based on binary Fredkin gates,
  - ***(3)*** Generalized Composition-Decomposition Methods
  - ***(4)*** Adaptations of **Ashenhurst/Curtis** decompositions.
  - ***(5)*** Levelized expansions for Shannon and Davio-like gates.
  - ***(6)*** adaptation of BDD-based, decomposition-based and technology mapping methods from standard binary logic

Our Past and Forthcoming Papers

(1) Multiple-valued reversible gates

(2) Multiple-valued quantum logic and synthesis methods

(3) Fuzzy reversible logic and synthesis methods

(4) **Ashenhurst/Curtis**-like decompositions of multi-valued reversible logic

(5) Levelized expansions for multi-valued Shannon and Davio-like gates.

(6) **Decision Diagrams** for binary and MV reversible logic

(7) *Genetic Algorithm* **combined with search for quantum logic**

(8) **Regular structures for MV unate, symmetric, threshold and other functions**

(9) **Visual Software**

End of Lecture 6