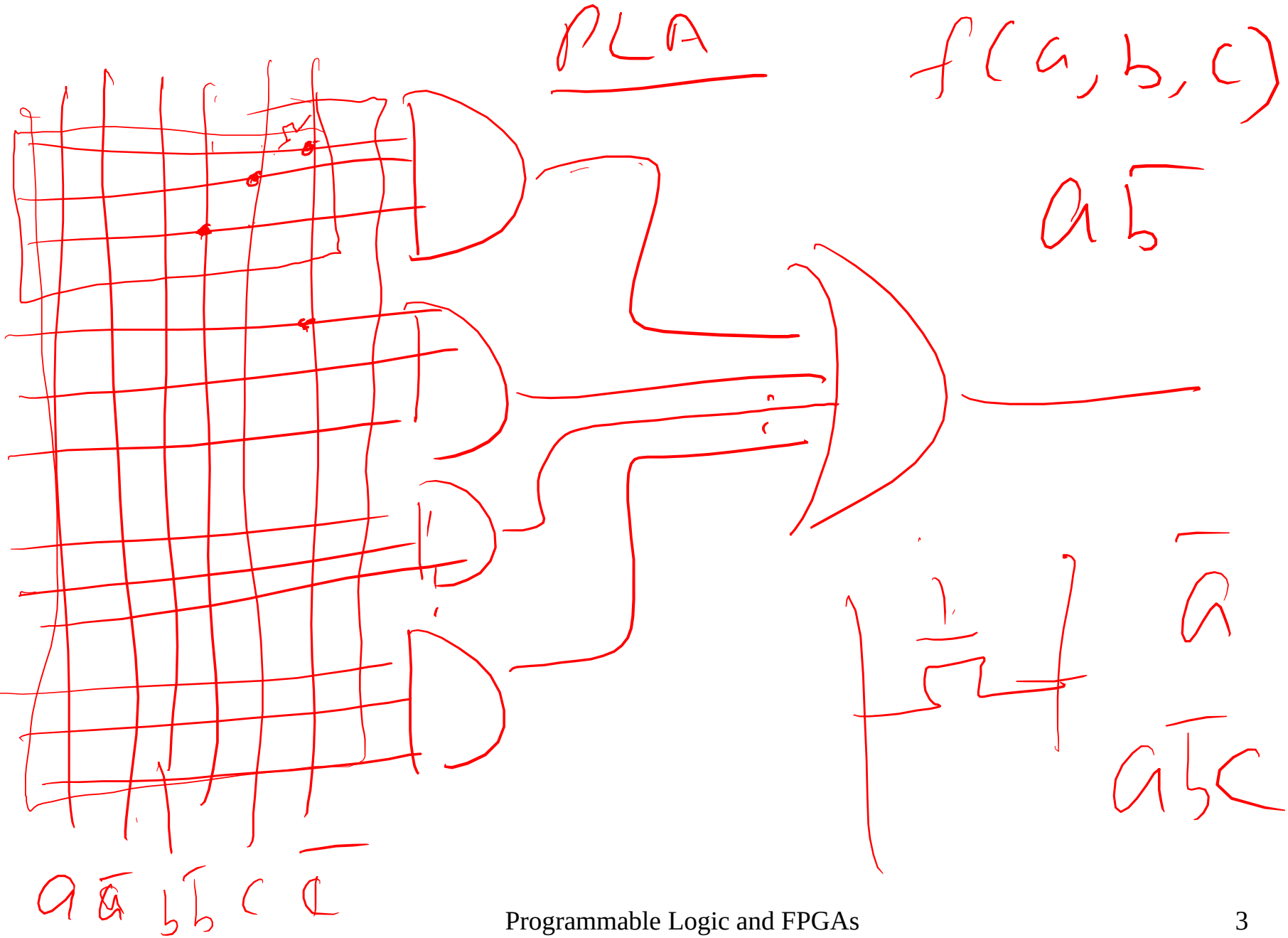
Lecture 32: Programmable Logic

M. Balakrishnan

Logic Implementation Options

To implement a n-variable function (with k minterms)

- n to 2^n decoder + k input OR gate
- 2^n:1 multiplexer
- $2^{(n-1)}$:1 multiplexer + 1 inverter
- $2^n \times 1$ ROM
- $n \times p \times 1$ PLA with $p \leq k$
- FPGA


PLA
f(a, b, c)
$a\bar{b}$
$\bar{a}$
$a\bar{b}c$
$a\ \bar{a}\ b\ \bar{b}\ c\ \bar{c}$

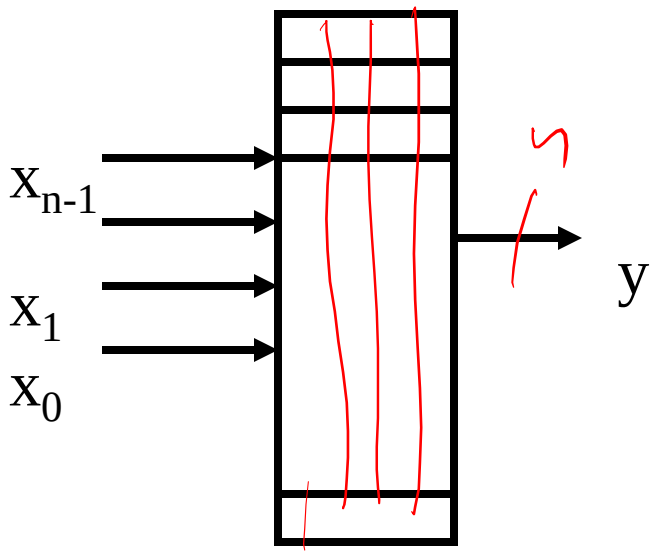
ROM (Read Only Memory)

$2^n \times m$ ROM (n address and m outputs)

LUT

n

2^n

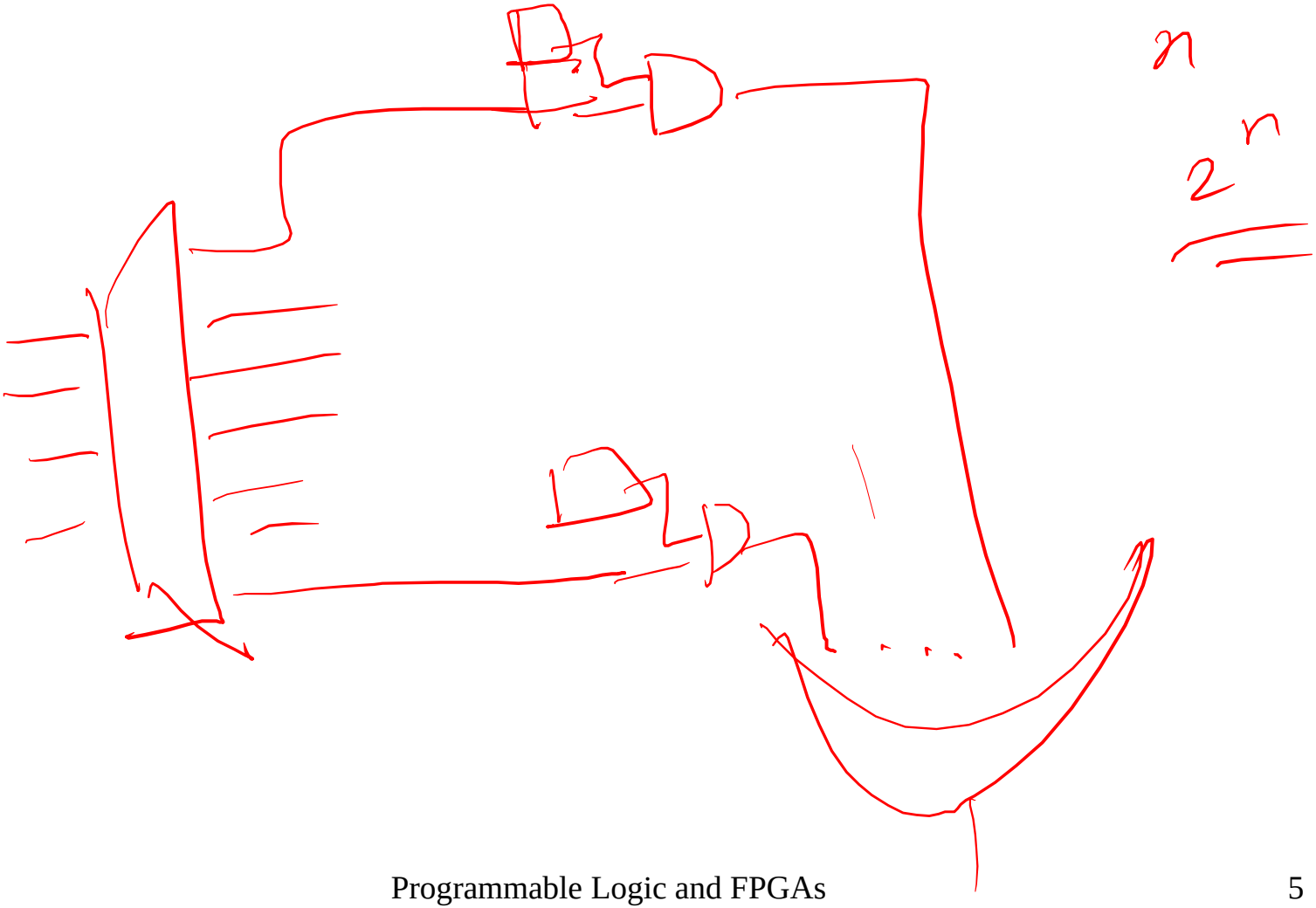


16X4

a_0 a_1 a_2 a_3

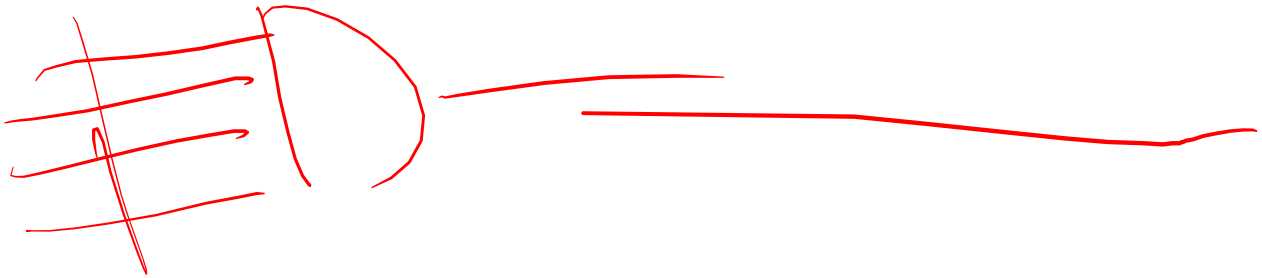
16X1

ROM Design Example



Implementing Logic Using ROMs

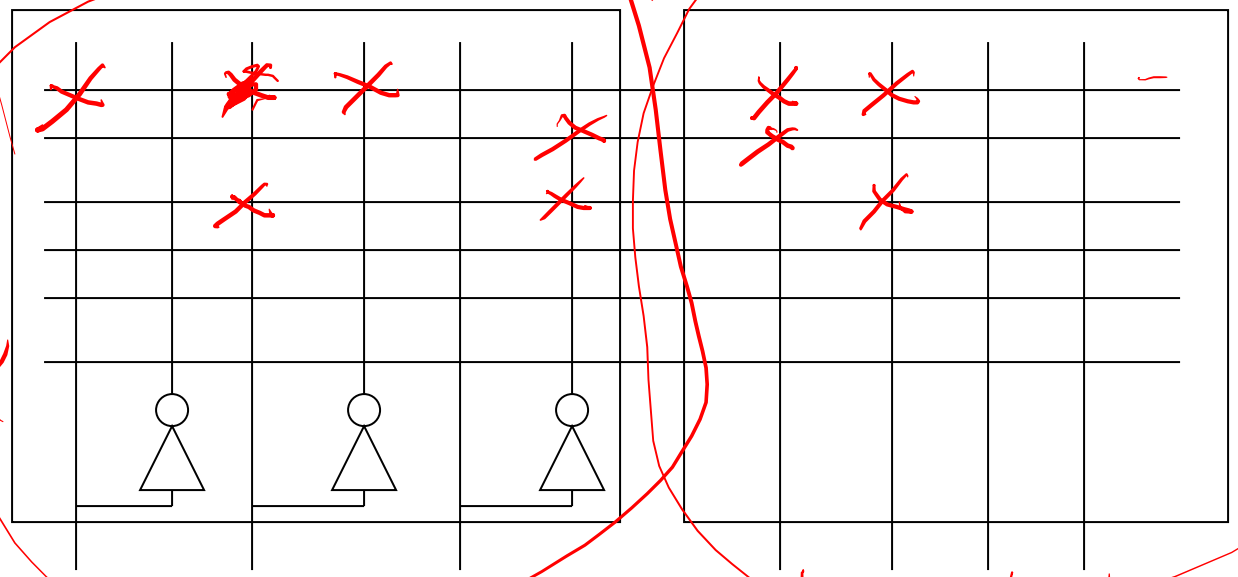
- Direct implementation of the function
- Extremely flexible as reprogramming can implement a completely different function



PLA (Programmable Logic Arrays)

PLA Specification: n × k × m

(N inputs, K product terms and M outputs)



Implementing Logic Using PLAs

NAND NOND

- Direct implementation of SoPs
- Implement product terms in the AND plane
- Implement sum in the OR plane

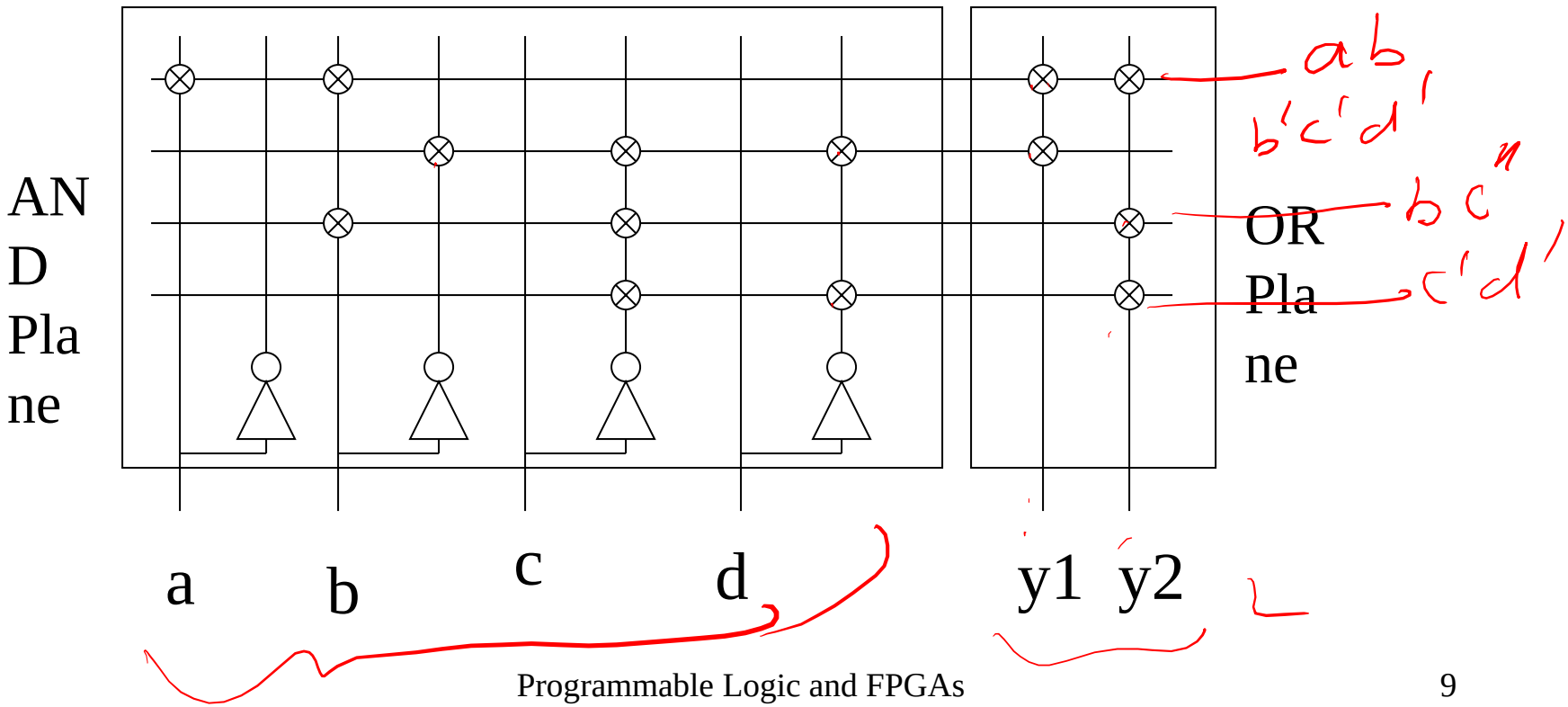
K AND gates with 2n inputs — $\boxed{K \times 2n}$

m OR gates with K inputs — $\boxed{m \times K}$

Implementing Logic Using PLAs (Example)

1 2 1 3 4

y1= ab + b'c'd' y2 = ab + bc' + c'd'



Programmable Devices

- Prefabricated Silicon
- Logic implemented by programming the basic cells and the interconnect
- Very fast *turnaround* time
- Limited design flexibility
- Low development *time* and *cost*

ASIC

Why FPGA?

ASIC

Custom logic implemented as ASICs have the following merits and demerits

Pros:

a. Reduced system complexity.

b. Improved performance.

Cons:

a. Very expensive to develop.

b. Delay introduction of product to market (time to market) because of increased design time.

Why FPGA (contd.)?

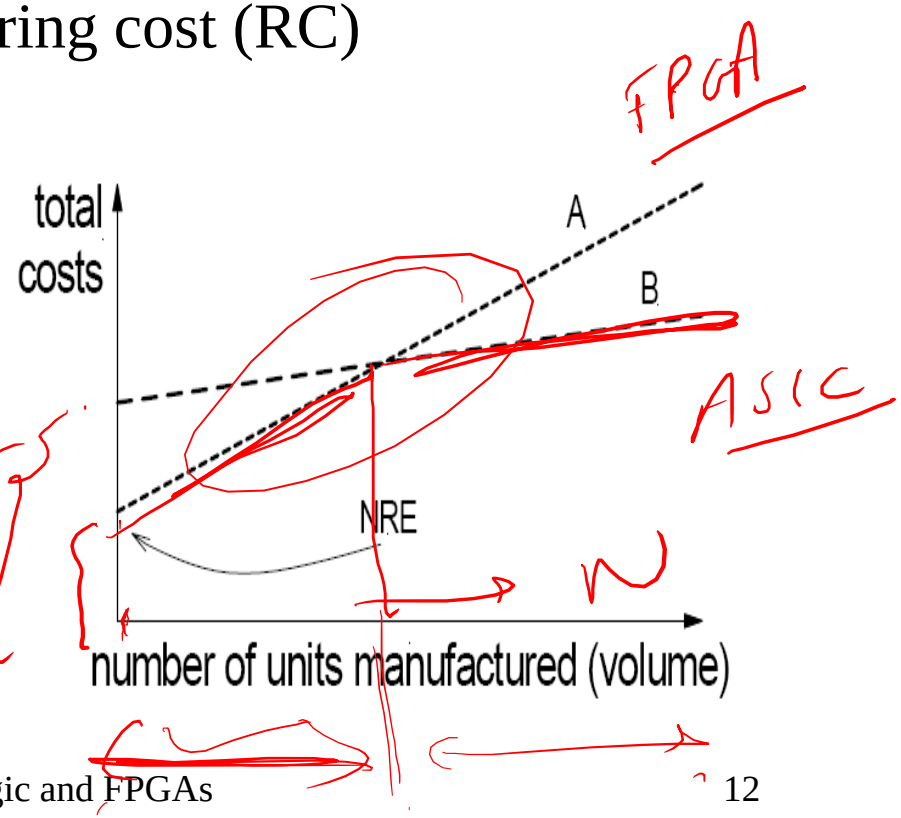
Need to worry about two kinds of costs:

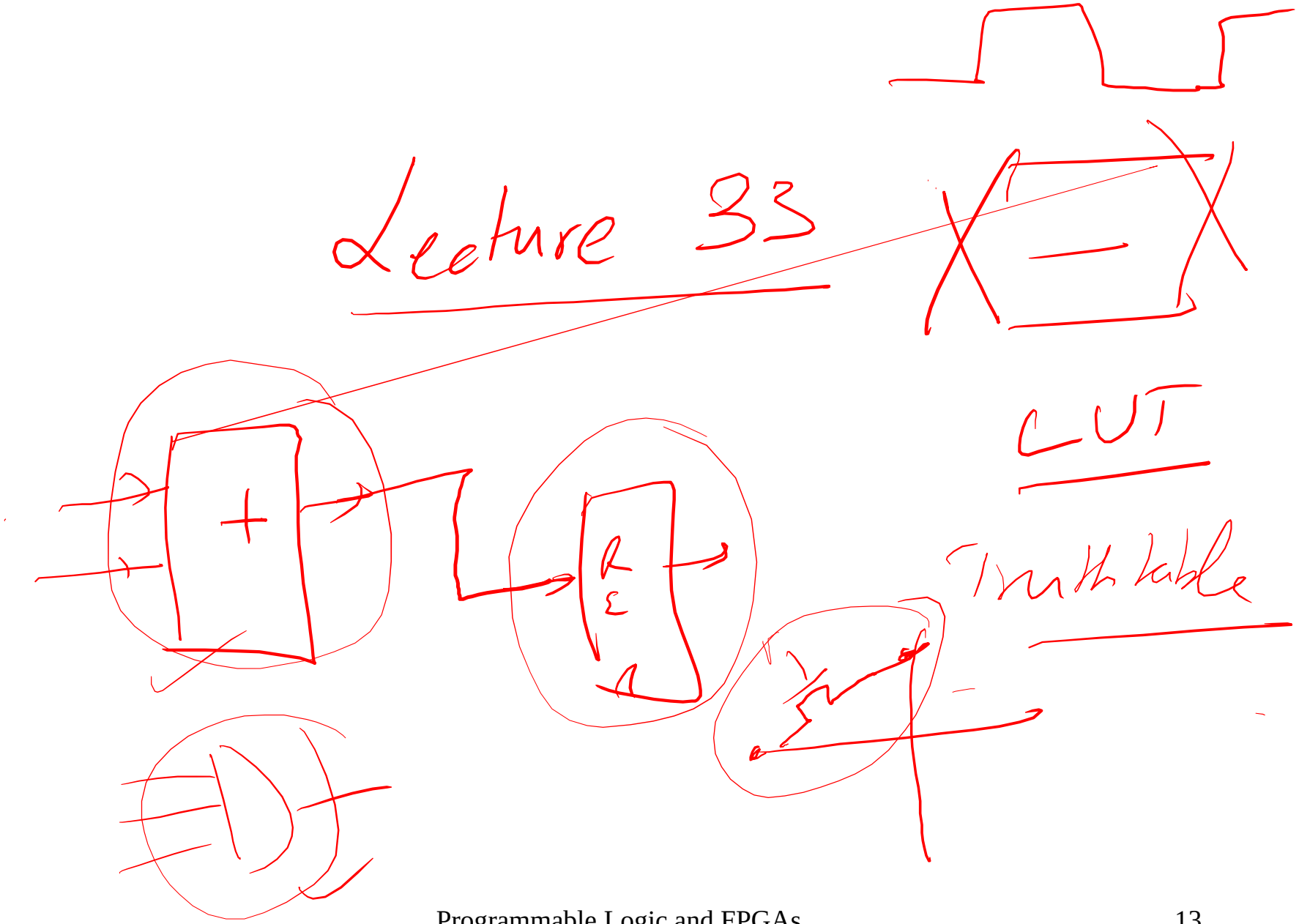
a. Cost of development, sometimes called non-recurring engineering (NRE)

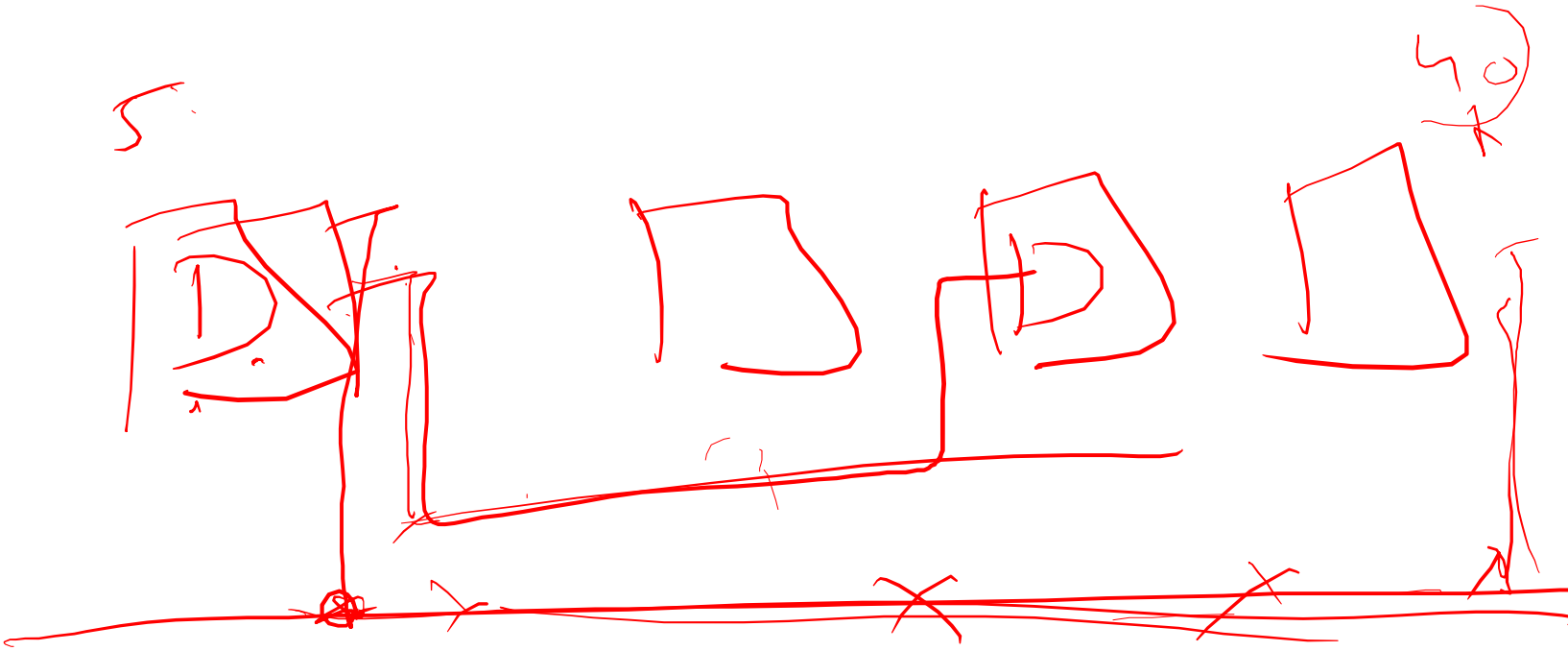
b. Cost of manufacture or Recurring cost (RC)

A: FPGAs (low NRE, high RC)

B: ASICs (high NRE, low RC)




Lecture 33
LUT
Truth table

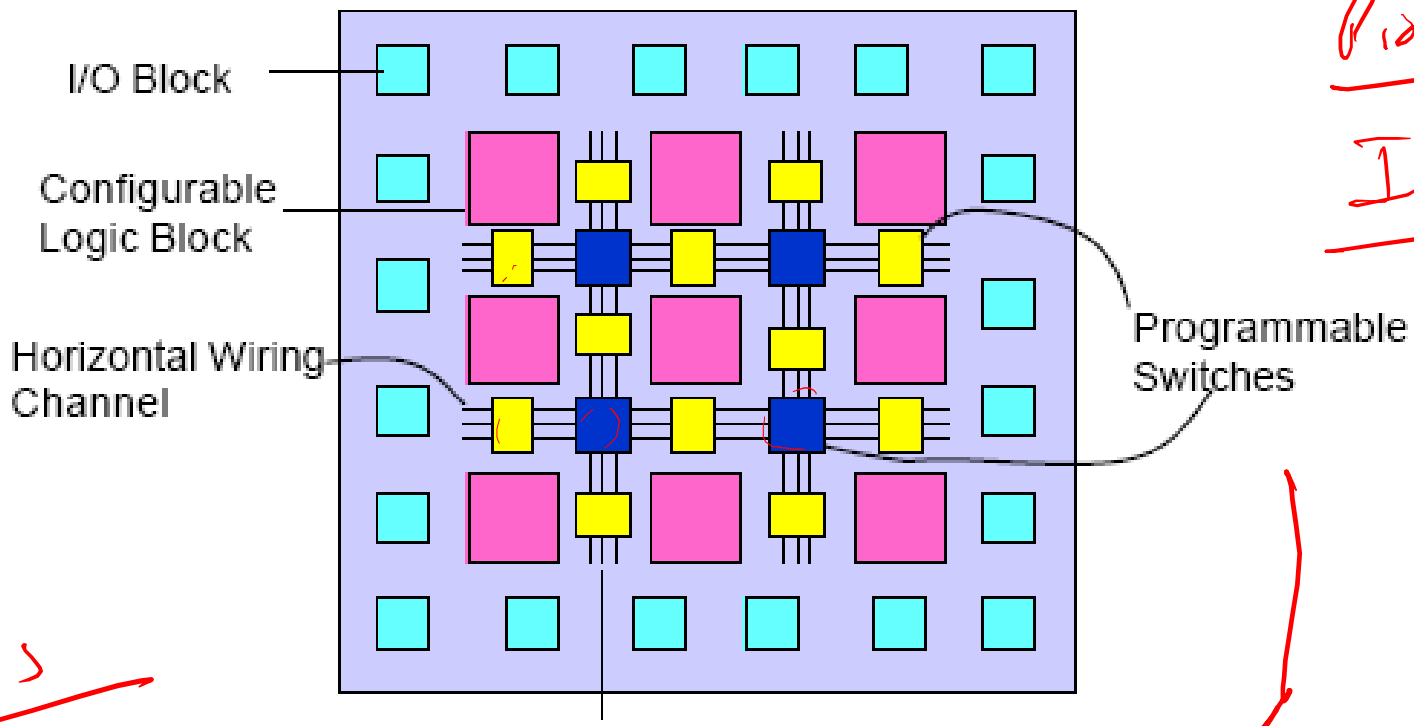


XILINX

FPGA Blocks

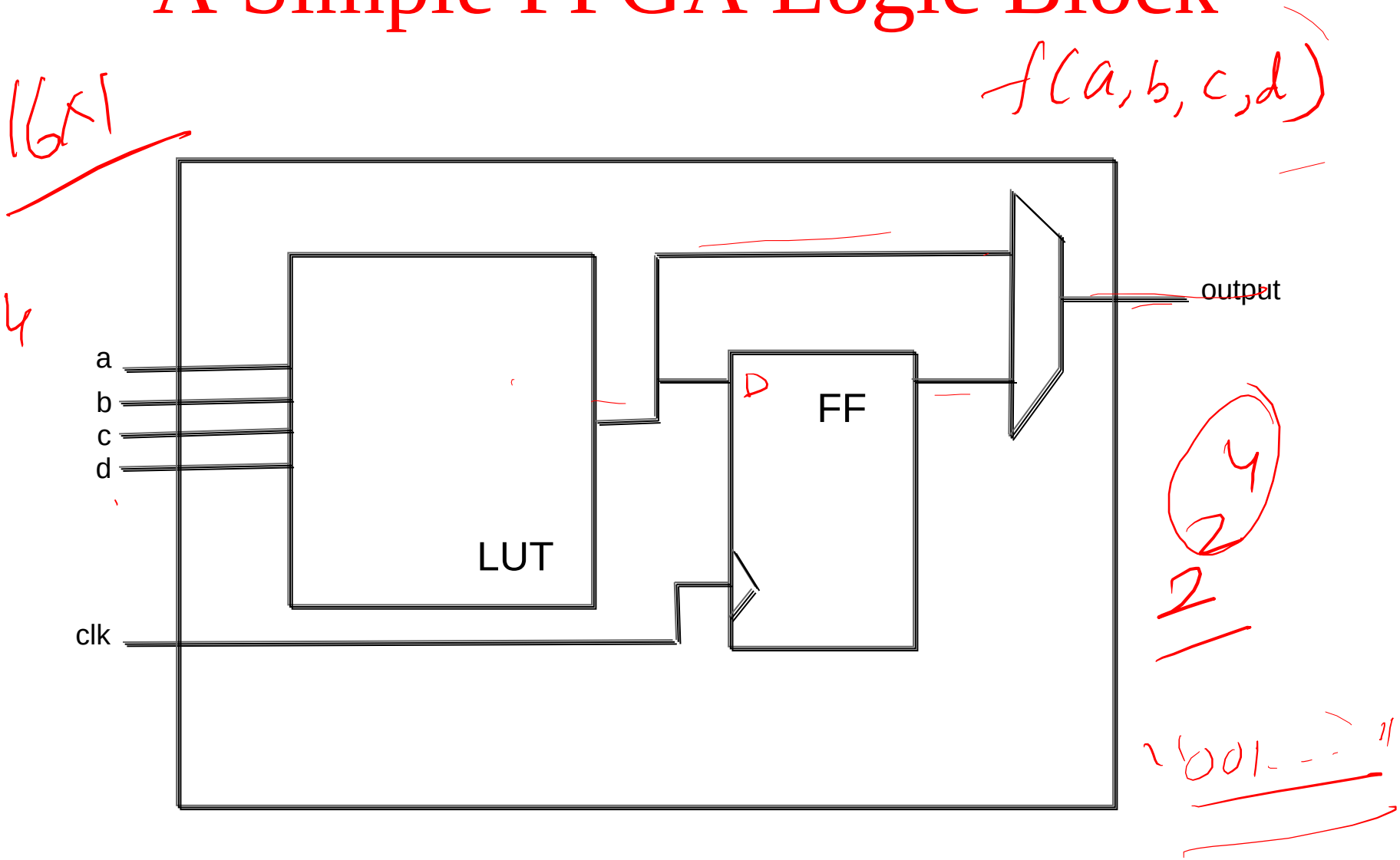
Pins

IOB

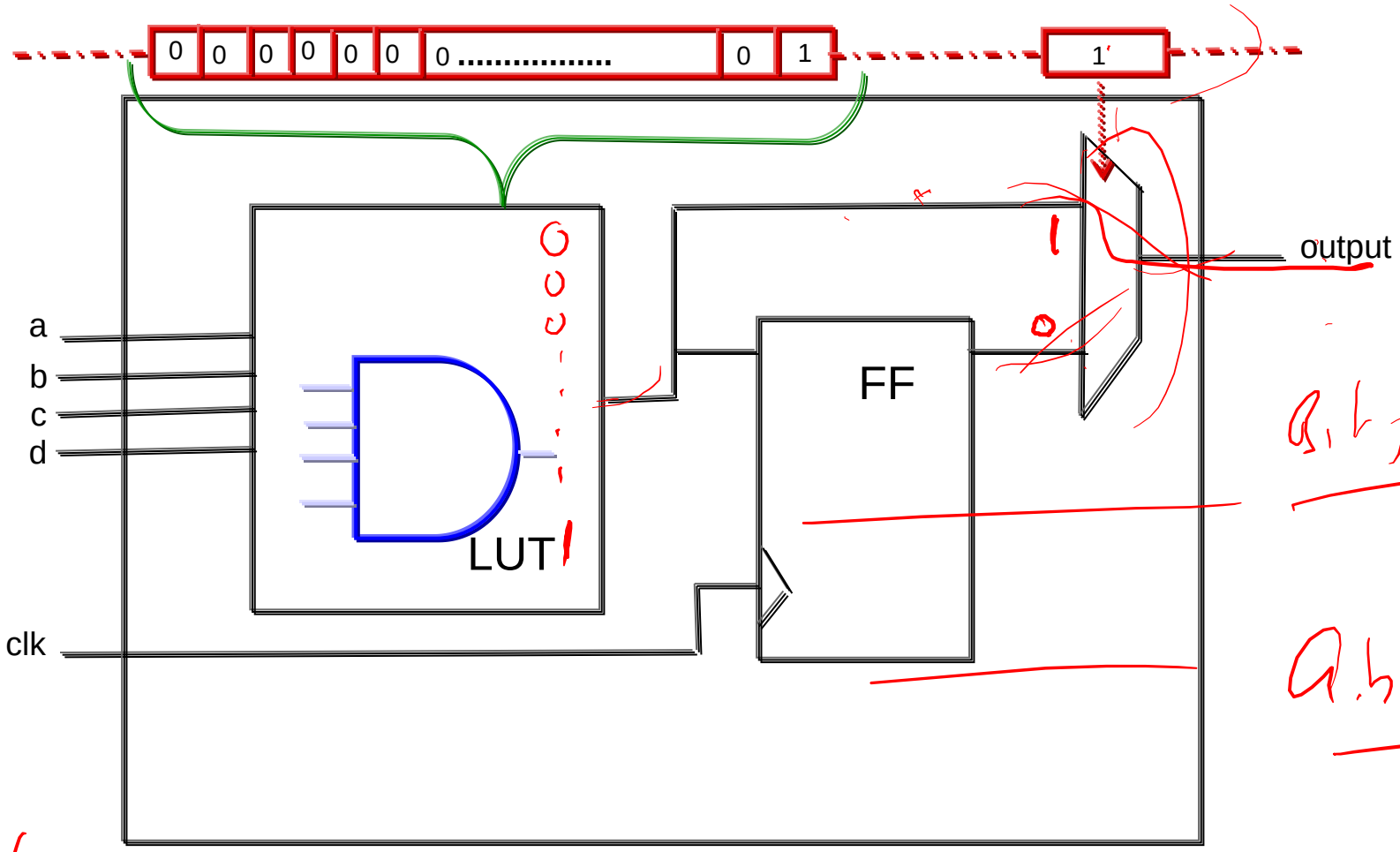


CLBs

A Simple FPGA Logic Block



A Simple FPGA Logic Block



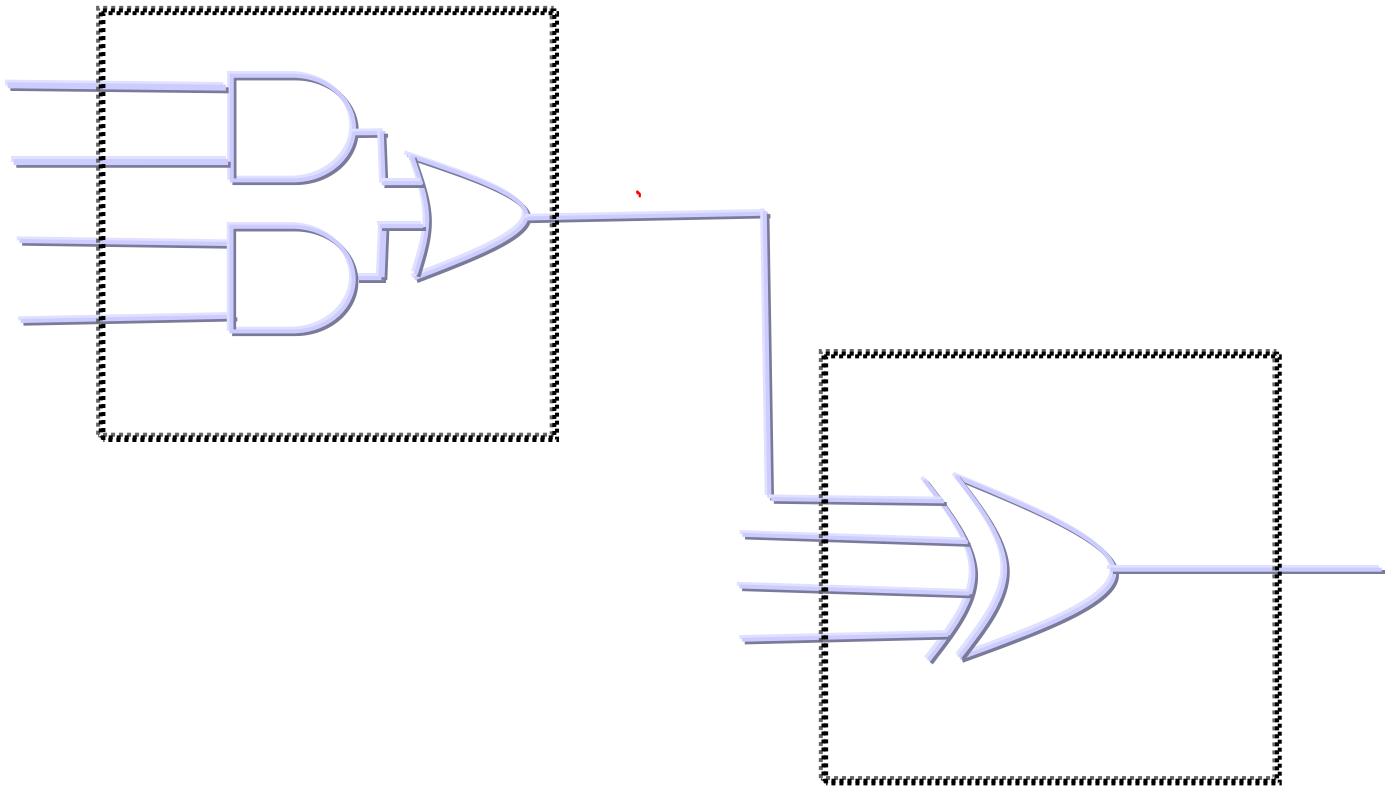
Bitfile

a.b.c.d

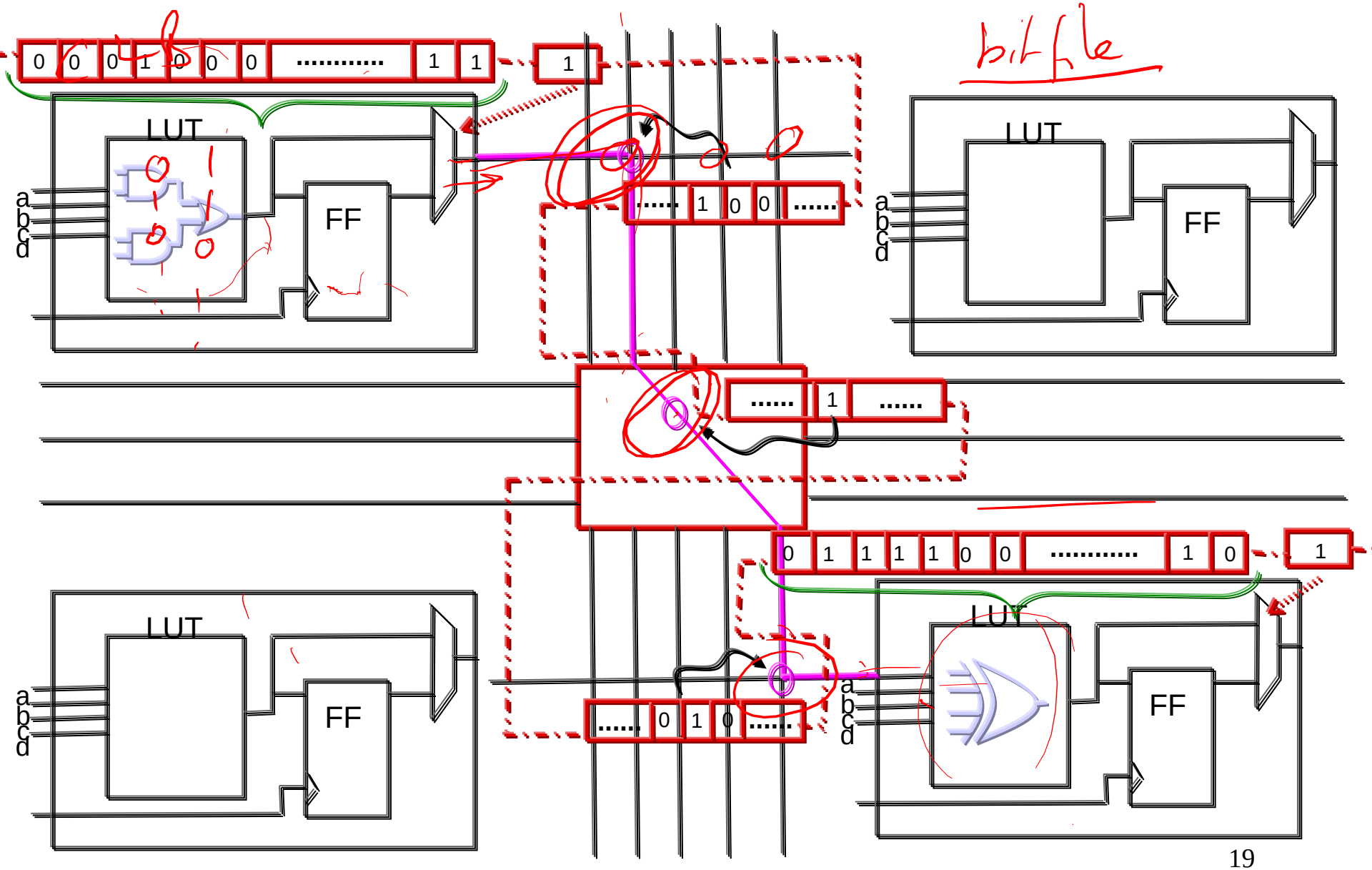
MUX

Decoder

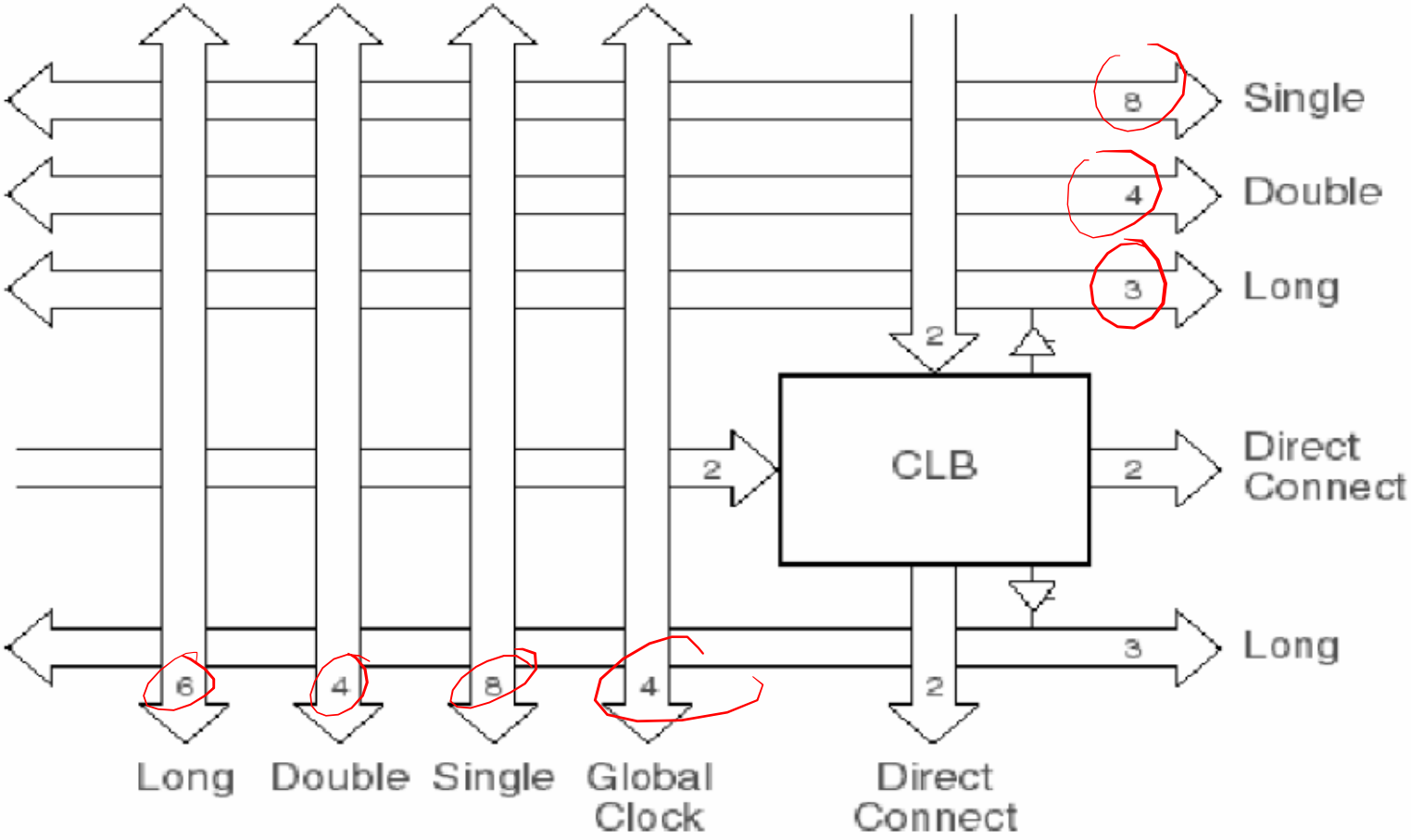
Simple Circuit



LUT Configuration bits



Interconnections

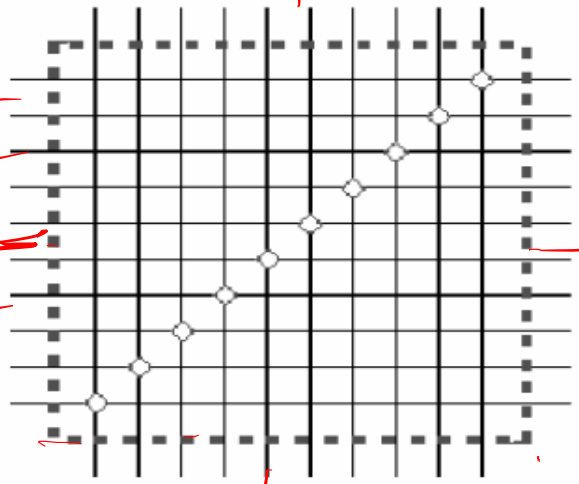


Source: xilinx.com

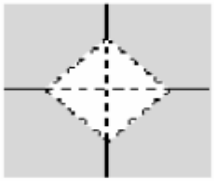
Programmable Interconnects

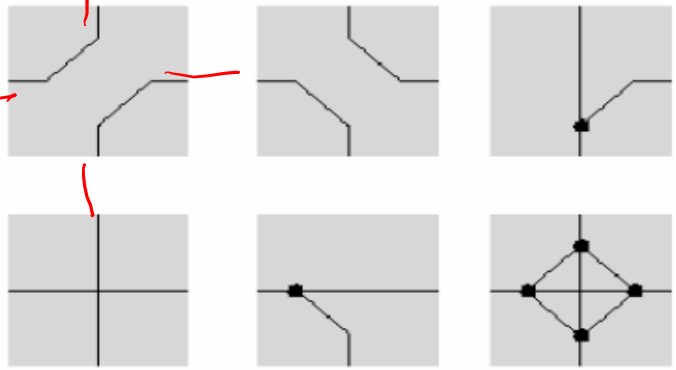
- **Connection box**
  - Connects input/output of logic block to interconnect channels.
- **Switch box**
  - Enables the connection of two interconnect lines.
- **Transmission gate (or a pass transistor) is used for each connection.**

Programmable Switching Matrix



programmable switch element





turning the corner, etc.

Methods of Interconnection

Direct Interconnect

- Connects adjacent CLBs

General Purpose Interconnection

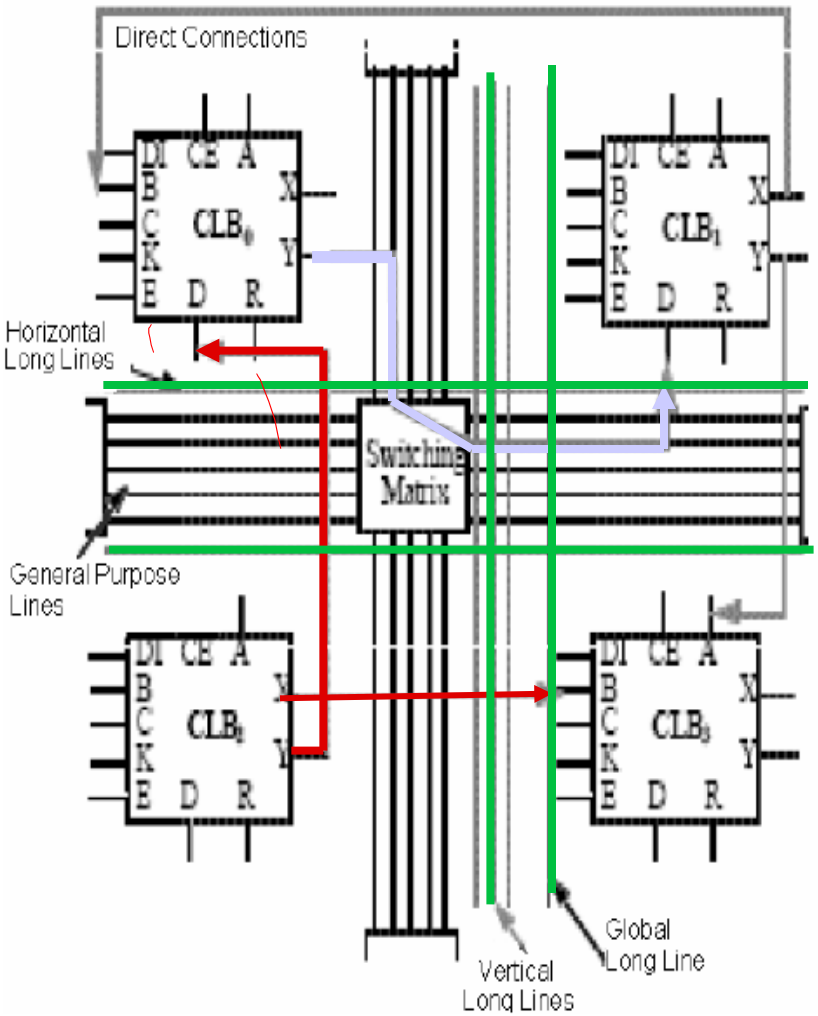
- Connects general interconnections and passes through one or more switch boxes ➔ slow

Long line interconnect through direct interconnects without going through switch boxes ➔ very fast

-Time critical signals

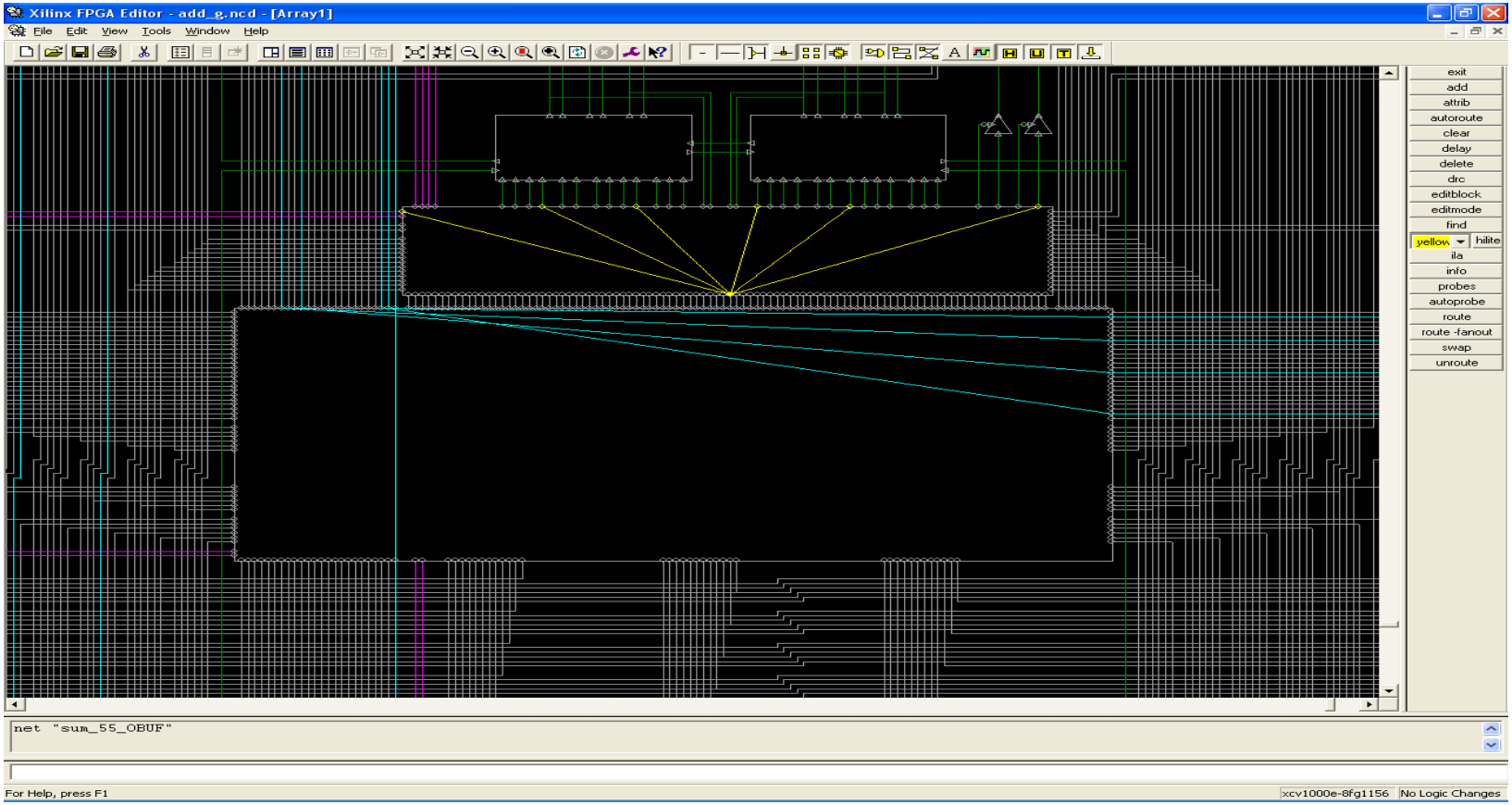
-Horizontal, vertical & Global long lines

-Global long lines for clocks and resets



Source: xilinx.com

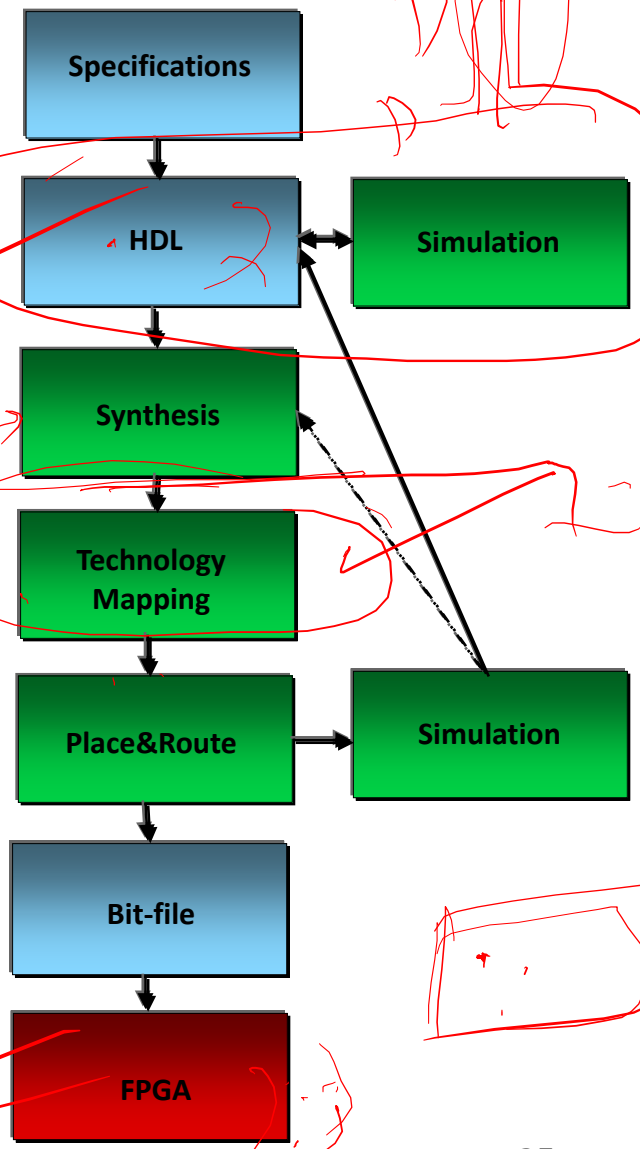
Programmable Interconnects



Snapshot from FPGA Editor

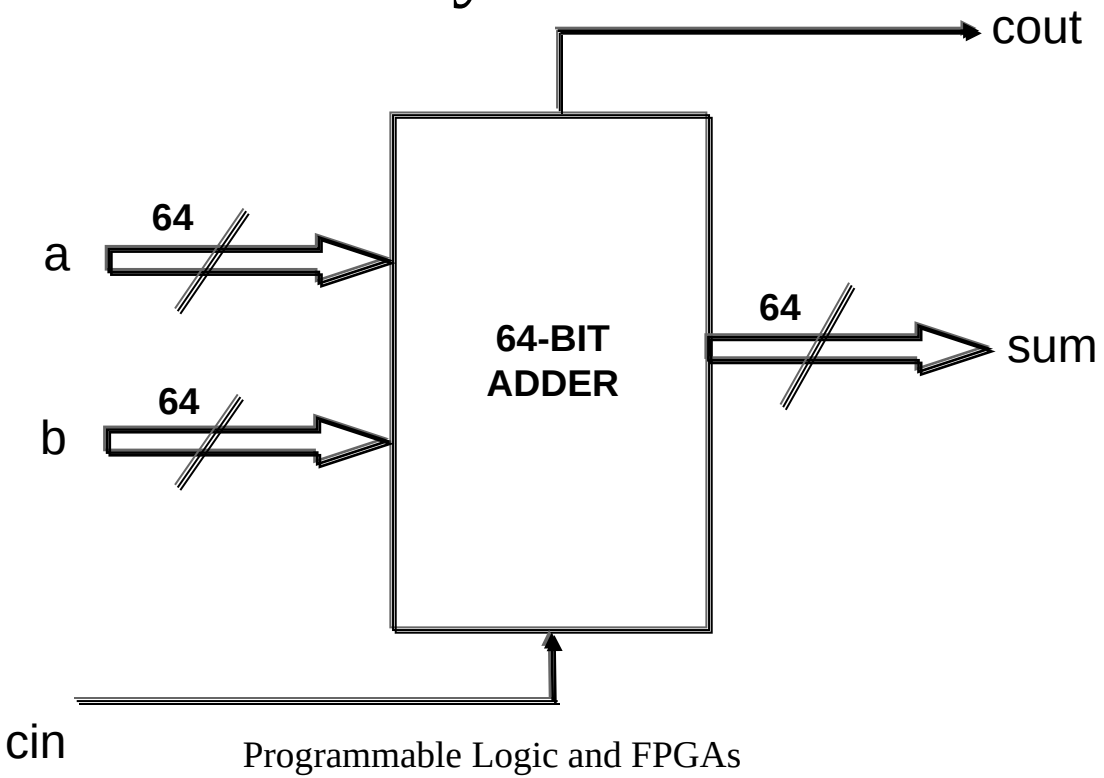
FPGA Design Flow

- Specifications of Design.
- Converting into HDL.
- Synthesis steps (mostly automated)
  - ❑ Synthesize Design
  - ❑ Map design
  - ❑ Placing design inside FPGA
  - ❑ Routing design inside FPGA
  - ❑ Generate bit-stream

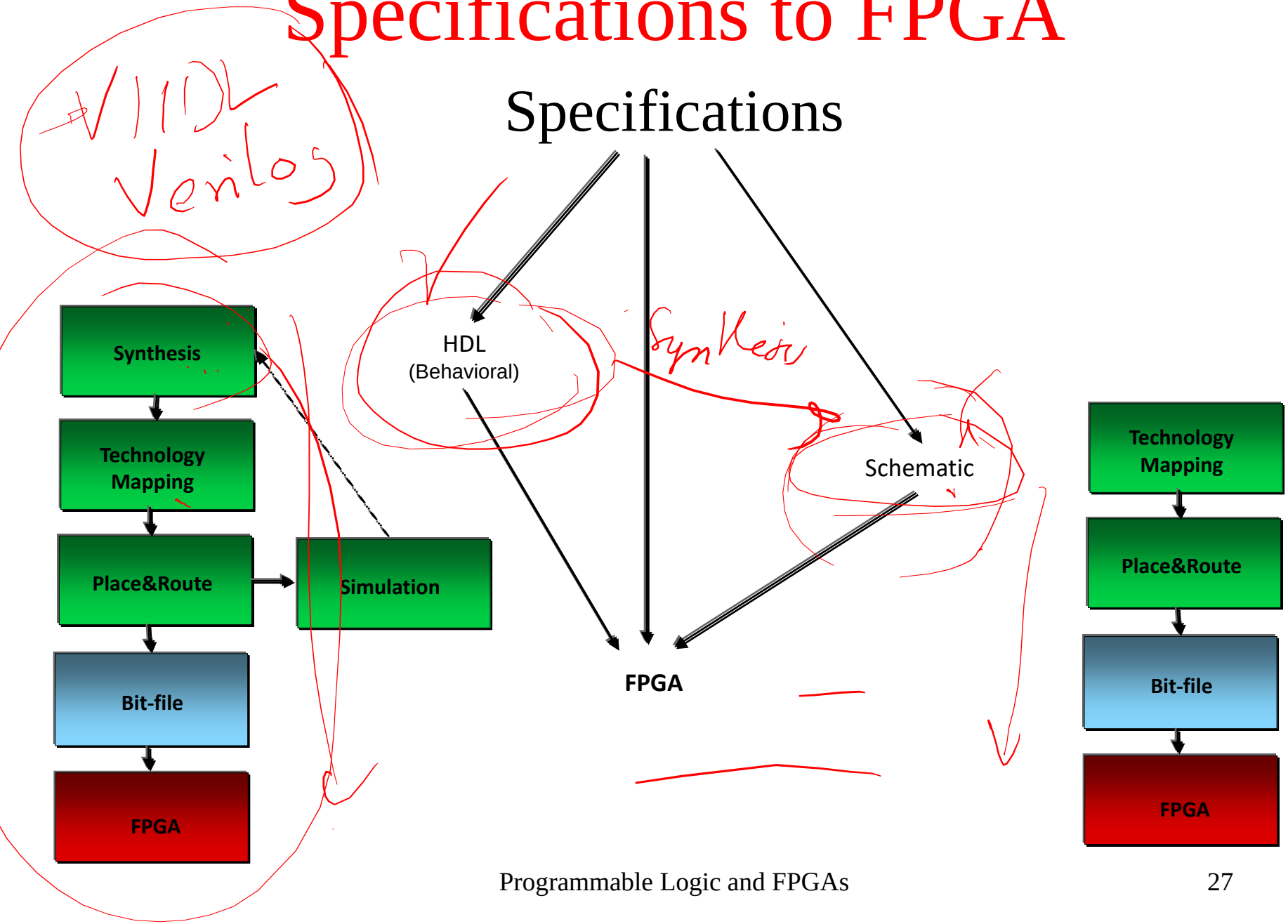


Specifications

- To add two 64-bit binary numbers with an additional carry in and generate a 64-bit output and 1-bit carry out.



Specifications to FPGA



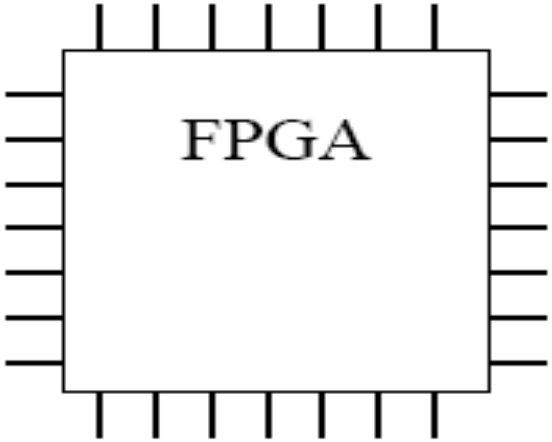
Specs (VHDL) to FPGA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity add_g is
generic (x   : natural := 63);

port (a : in  std_logic_vector (x downto 0);
b : in  std_logic_vector (x downto 0);
cin  : in  std_logic;
sum  : out std_logic_vector (x downto 0);
cout : out std_logic);
end entity add_g;

architecture behavior of add_g is
begin  -- behavior
adder: process(a,b,cin)
variable carry : std_logic;
variable isum : std_logic_vector(x downto
    0);
   begin
    carry := cin;
     for i in 0 to x loop
        isum(i) := a(i) xor b(i) xor carry;
        carry  := (a(i) and b(i)) or (a(i) and
        carry) or (b(i) and carry);
      end loop;
      sum  <= isum;
      cout <= carry;
 end process adder;
end architecture behavior;
```



Design Entry

Xilinx - ISE - D:\Gunjan\Xilinx8.2\adder_64\adder_64.ise - [adder.vhd]

File Edit View Project Source Process Window Help

Sources

Sources for: Synthesis/Implementation

- adder_64
- xcv1000e-8fg1156
  - add_g - behavior (adder.vhd)

Sources | Snapshots

Processes

Processes:

- Add Existing Source
- Create New Source
- View Design Summary
- Design Utilities
  - Create Schematic Symbo
  - View Command Line Log
  - View HDL Instantiation T
- User Constraints
- Synthesize - XST
  - View Synthesis Report
  - View RTL Schematic
  - View Technology Schem
  - Check Syntax
  - Generate Post-Synthesis
- Implement Design
  - Translate
  - Map
  - Place & Route
    - Place & Route Repor
    - Clock Region Report
    - Asynchronous Delay
    - Pad Report
    - Guide Results Repor
    - MPPR Results Utilitie
    - Generate Post-Place
    - View/Edit Placed De
    - View/Edit Routed De

Processes

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6
7  entity add_g is
8   generic (x    : natural := 63);
9      port (a    : in  std_logic_vector (x downto 0);
10           b    : in  std_logic_vector (x downto 0);
11           cin  : in  std_logic;
12           sum  : out std_logic_vector (x downto 0);
13           cout : out std_logic);
14 end entity add_g;
15
16 architecture behavior of add_g is
17 begin  -- behavior
18   adder: process(a,b,cin)
19            variable carry : std_logic;
20            variable isum : std_logic_vector(x downto 0);
21          begin
22            carry := cin;
23            for i in 0 to x loop
24              isum(i) := a(i) xor b(i) xor carry;
25              carry  := (a(i) and b(i)) or (a(i) and carry) or (b(i) and carry);
26            end loop;
27            sum  <= isum;
28            cout <= carry;
29            end process adder;
30 end architecture behavior;
31
32
```

Design Summary | adder

Transcript

```
Started : "Launching Technology Schematic Viewer for add_g.ngc".
```

Console | Errors | Warnings | Find in Files

Ln 7 Col 16 CAPS NUM SCRL VHDL

Snapshot from Xilinx ISE

Synthesizing the Design

- Synthesis : Optimization process of adapting a logic design to the logic resources available on the chip, like lookup tables, Long line, and dedicated carry.
- It means analyzing the whole design, and selecting which logic resources available in the FPGA will be used to perform the task.
- Gate level netlist is the output file.

Synthesis - Example

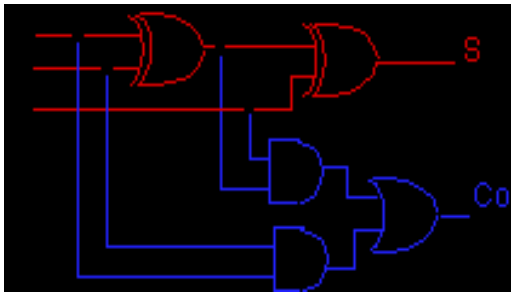
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity add_g is
generic (x   : natural := 63);

port (a : in  std_logic_vector (x downto 0);
b : in  std_logic_vector (x downto 0);
cin  : in  std_logic;
sum  : out std_logic_vector (x downto 0);
cout : out std_logic);
end entity add_g;

architecture behavior of add_g is
begin  -- behavior
adder: process(a,b,cin)
variable carry : std_logic;
variable isum : std_logic_vector(x downto
    0);
   begin
    carry := cin;
     for i in 0 to x loop
        isum(i) := a(i) xor b(i) xor carry;
        carry  := (a(i) and b(i)) or (a(i) and
        carry) or (b(i) and carry);
      end loop;
      sum  <= isum;
      cout <= carry;
 end process adder;
end architecture behavior;
```

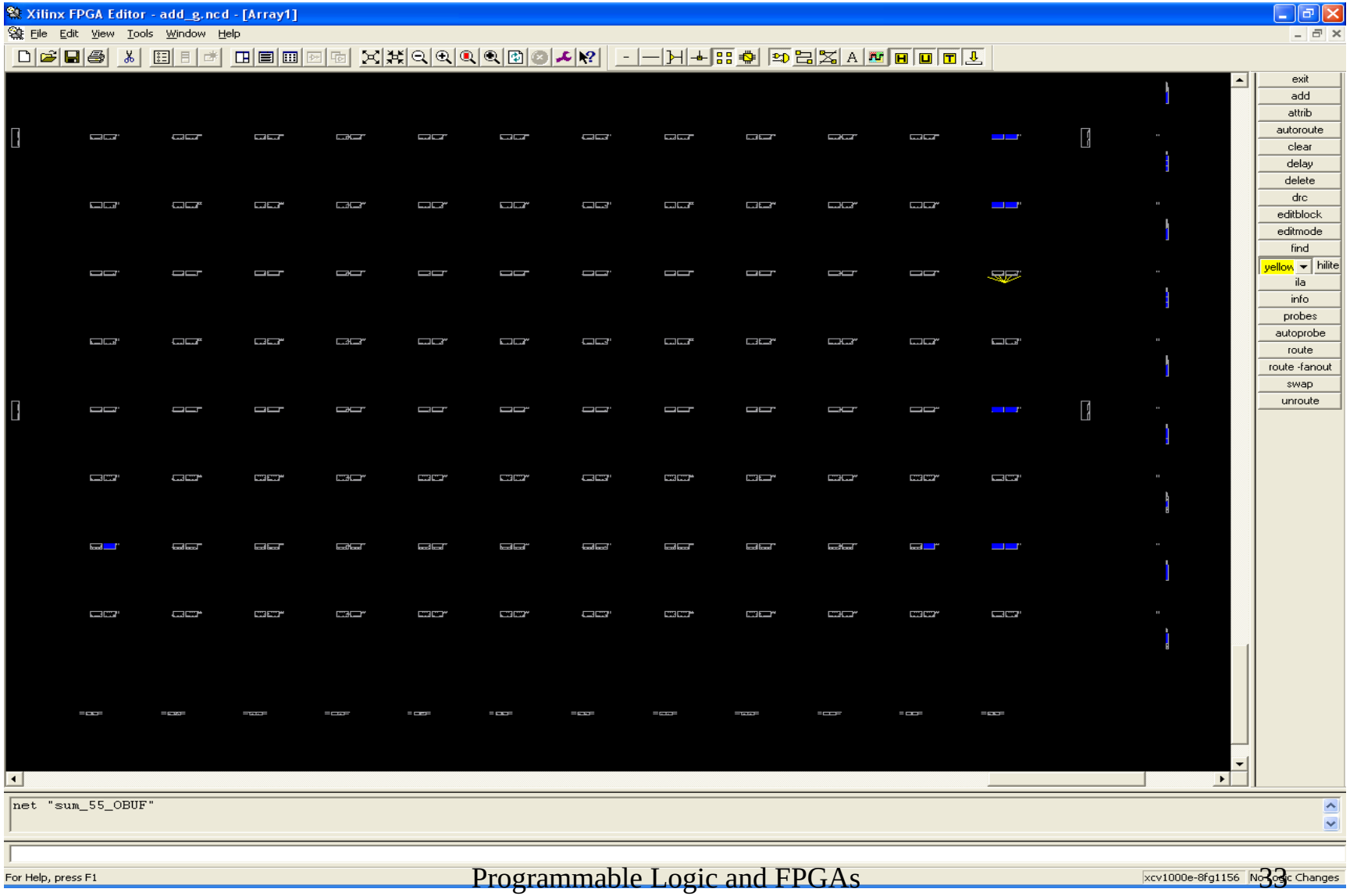
Synthesis



Mapping the Design

- Mapping : Process of assigning portions of the logic design to the physical chip resources (CLBs).
- Function similar to synthesizing.
- Synthesis is not necessarily specific to a particular FPGA, but mapping usually is.

Mapping the Design (contd.)

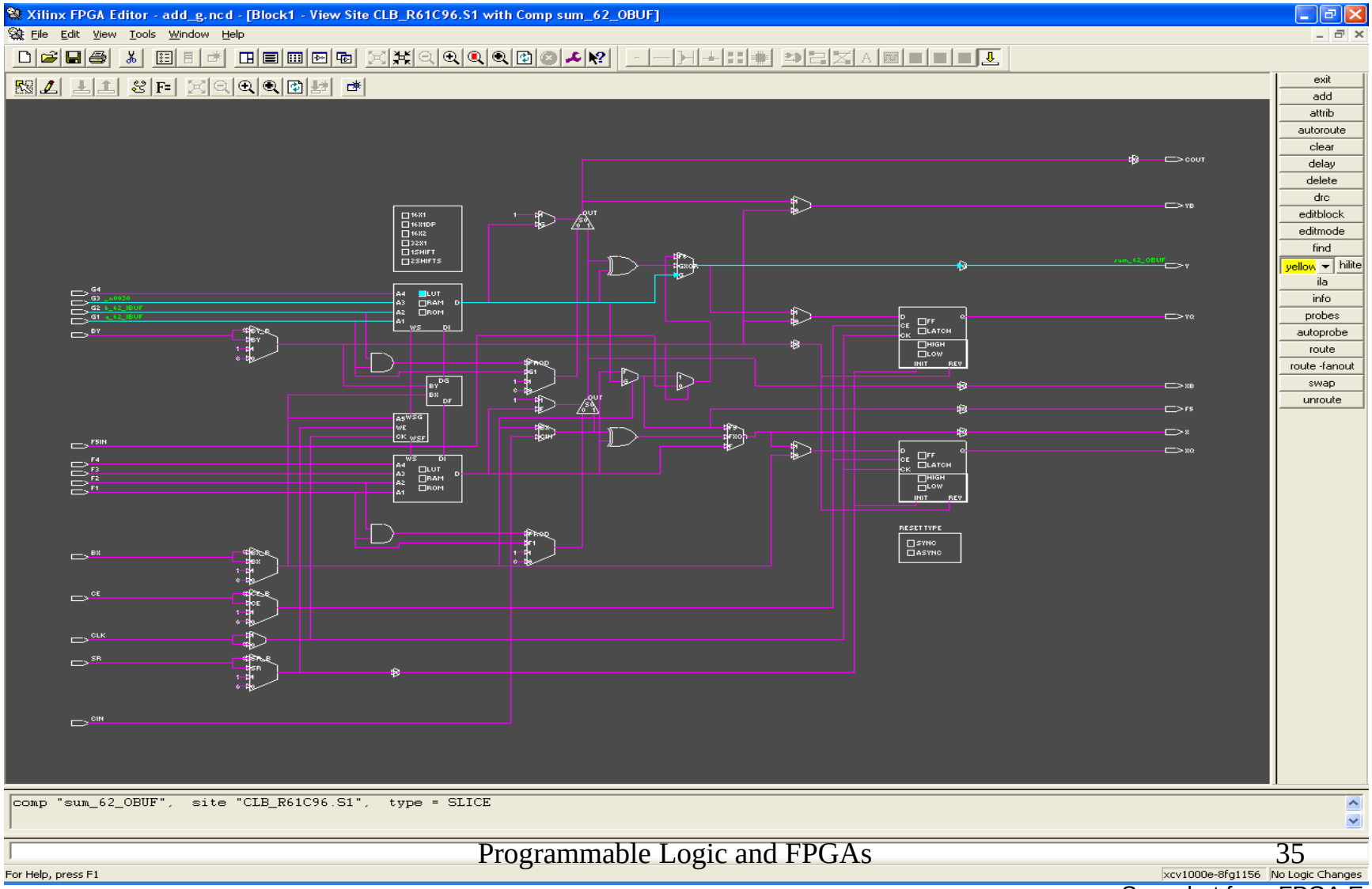


Snapshot from FPGA Editor

Placing the Design

- Placing : In FPGAs, the process of assigning specific parts of the design to specific locations (CLBs) on the chip.
- Usually done automatically.
- Once the design has been converted into the logic resources of the FPGA, we find a location for these within the FPGA. (Generally a 2-dimensional grid structure)

Placing the Design (contd.)

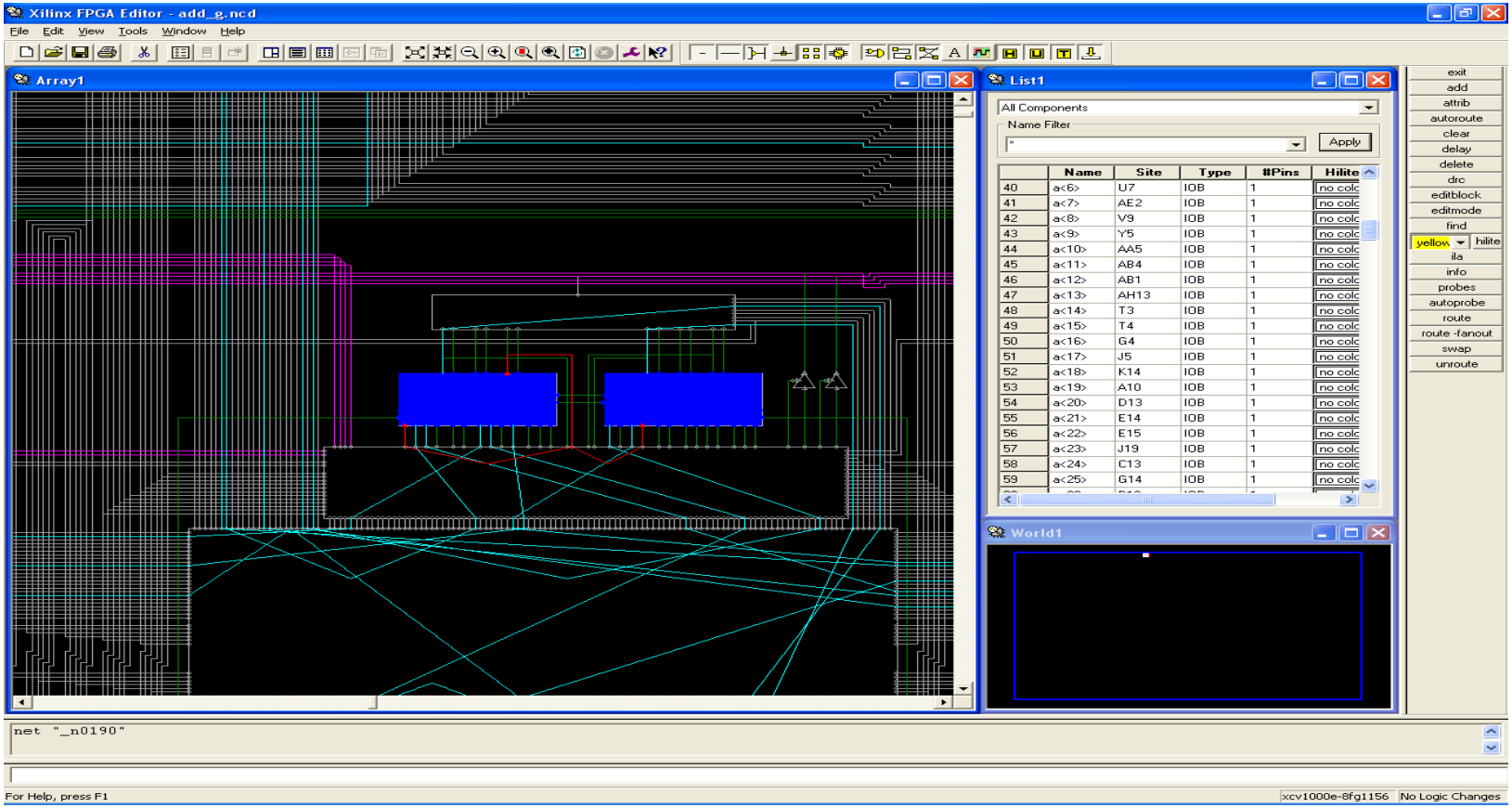


Snapshot from FPGA Editor

Routing the Design

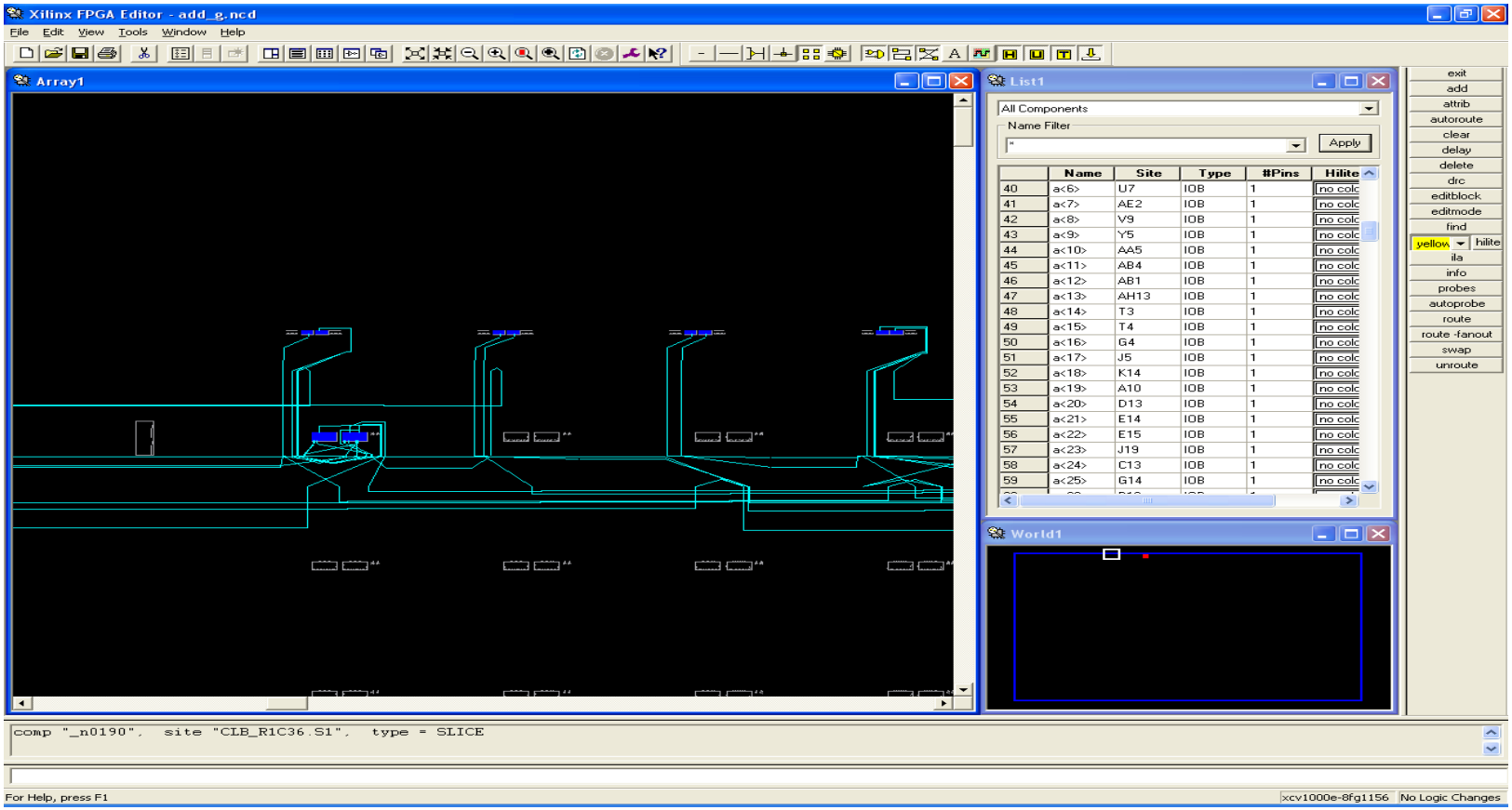
- Routing : The process of creating the desired interconnection of logic cells to make them perform the desired function.
- Routing follows after placement.
- Once logic resources have been assigned a location within the FPGA, we need to interconnect the logic resources using internal buses inside the FPGA.

Routing the Design (contd.)



Snapshot from FPGA Editor

Routing the Design (contd.)



Snapshot from FPGA Editor

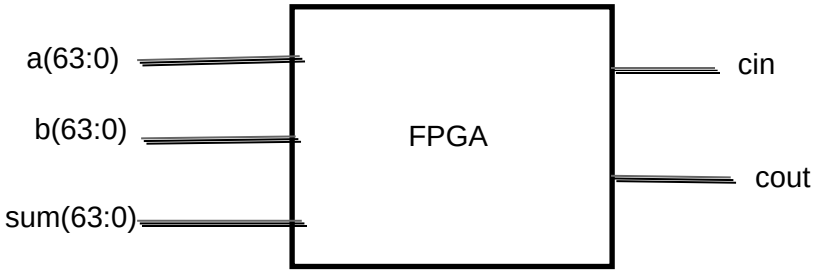
UCF

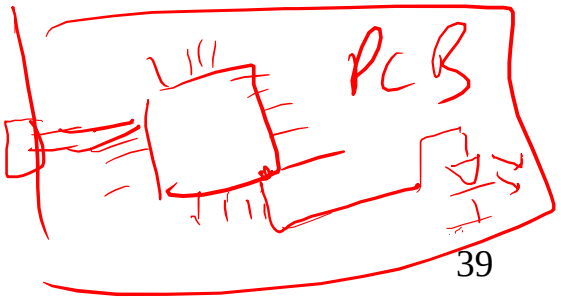
Assigning Pins

When implementing an entity in FPGA, the input and output ports are mapped to pins of the FPGA

```
entity add_g is
generic (x   : natural := 63);

port (a : in  std_logic_vector (x downto
0);
b : in  std_logic_vector (x downto 0);
cin  : in  std_logic;
sum  : out std_logic_vector (x downto 0);
cout : out std_logic);
end entity add_g;
```





Assigning Pins (cont.)

- A file called a UCF (User Constraint File) is used to define which pin will be connected to a particular input or output.

- Within Xilinx Project manager, the “assign package pin” function can be used to easily define input and output pin location.

Example UC File

```
entity add_g is
generic (x   : natural := 63);

port (a : in  std_logic_vector (x downto 0);
b : in  std_logic_vector (x downto 0);
cin  : in  std_logic;
sum  : out std_logic_vector (x downto 0);
cout : out std_logic);
end entity add_g;
```

```
NET "cin"   LOC = "T9";
NET "cout"  LOC = "M13" ;
```

Convert Design into Bit-stream

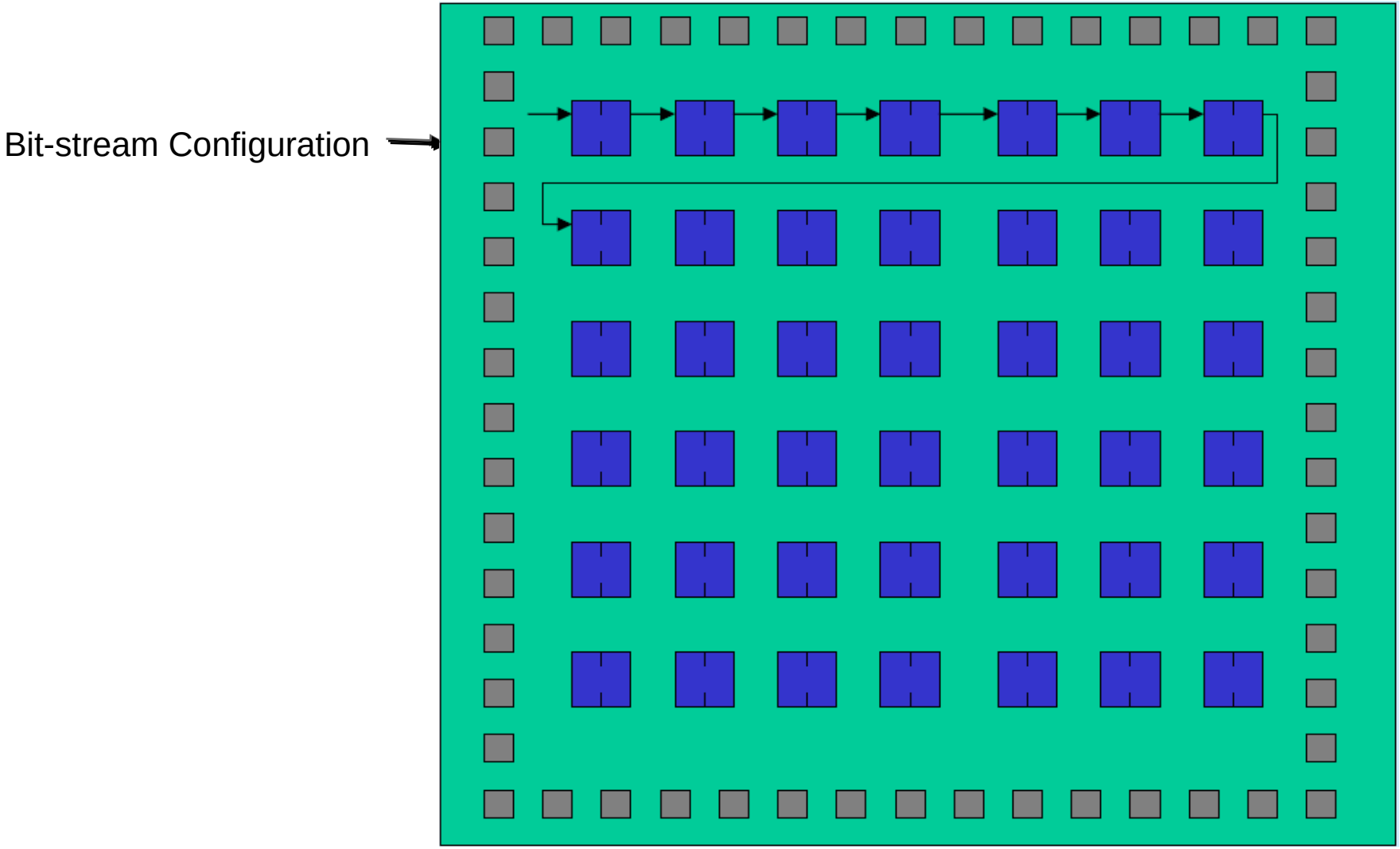
100K

- Always done using automated tools.

- The placed and routed design is converted into a bit-stream that is downloaded into the FPGA to configure it.

slow tools

Design to Bit-stream (contd.)



Design Summary

Xilinx - ISE - D:\Gunjan\Xilinx8.2\adder_64\adder_64.ise - [Design Summary]

File Edit View Project Source Process Window Help

Sources

Sources for: Synthesis/I

adder_64

xcv1000e-8fg1156

add_g - beha

Processes

Processes:

View Synt

View RTL

View Tec

Check Sy

Generate

Implement De

Translate

Map

Place & R

Place

Clock

Async

Pad F

Guide

MPPF

Gene

View

View

Analy

Gene

Gene

Gene

Back

Generate Prog

Programm

Generate

Configure

Update Bitstre

FPGA Design Summary

Design Overview

Summary

IOB Properties

Timing Constraints

Pinout Report

Clock Report

Errors and Warnings

Synthesis Messages

Translation Messages

Map Messages

Place and Route Messages

Timing Messages

Bitgen Messages

All Current Messages

Detailed Reports

Synthesis Report

Translation Report

Map Report

Place and Route Report

Static Timing Report

Bitgen Report

Project Properties

Enable Enhanced Design Summ

Enable Message Filtering

Display Incremental Messsages

Enhanced Design Summary Contents

Show Errors

Show Warnings

Show Failing Constraints

Show Clock Report

ADDER_64 Project Status

Project File:	adder_64.ise	**Current State:**	Placed and Routed
Module Name:	add_g	**• Errors:**	No Errors
Target Device:	xcv1000e-8fg1156	**• Warnings:**	1 Warning
Product Version:	ISE, 8.1i	**• Updated:**	Thu Feb 12 07:26:10 2009

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	129	24,576	1%	
Logic Distribution				
Number of occupied Slices	96	12,288	1%	
Number of Slices containing only related logic	96	96	100%	
Number of Slices containing unrelated logic	0	96	0%	
Total Number of 4 input LUTs	129	24,576	1%	
Number of bonded IOBs	194	660	29%	
Total equivalent gate count for design	774			
Additional JTAG gate count for IOBs	9,312			

Performance Summary

Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints		

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Wed Feb 11 23:42:32 2009	0	0	0
Translation Report	Current	Thu Feb 12 07:26:00 2009	0	0	0
Map Report	Current	Thu Feb 12 07:26:04 2009	0	0	2 Infos
Place and Route Report	Current	Thu Feb 12 07:26:07 2009	0	1 Warning	1 Info
Static Timing Report	Current	Thu Feb 12 07:26:10 2009	0	0	2 Infos
Bitgen Report					

Design Summary | adder

Transcript

```
Started : "Launching Technology Schematic Viewer for add_g.ngc".
```

Console | Errors | Warnings | Find in Files

Snapshot from Xilinx ISE

Synthesis Report

Xilinx - ISE - D:\Gunjan\Xilinx8.2\adder_64\adder_64.ise - [Design Summary]

```
TABLE OF CONTENTS
  1) Synthesis Options Summary
  2) HDL Compilation
  3) HDL Analysis
  4) HDL Synthesis
     4.1) HDL Synthesis Report
  5) Advanced HDL Synthesis
     5.1) Advanced HDL Synthesis Report
  6) Low Level Synthesis
  7) Final Report
     7.1) Device utilization summary
     7.2) TIMING REPORT

=========================================================================
*                      Synthesis Options Summary                        *
=========================================================================
---- Source Parameters
Input File Name                    : "add_g.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "add_g"
Output Format                      : NGC
Target Device                      : xcv1000e-8-fg1156

---- Source Options
Top Module Name                    : add_g
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
FSM Style                          : lut
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : YES
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
XOR Collapsing                     : YES
ROM Style                          : Auto
Mux Extraction                     : YES
Resource Sharing                   : YES
Multiplier Style                   : lut
Automatic Register Balancing       : No
```

Started : "Launching Design Summary".

Snapshot from Xilinx ISE

Synthesis Report – Device Utilization Summary

Xilinx - ISE - D:\Gunjan\Xilinx8.2\adder_64\adder_64.ise - [Design Summary]

File Edit View Project Source Process Window Help

Sources

Sources for: Synthesis/I

adder_64

xcv1000e-8fg1156

add_g - beha

Processes

Processes:

User Constrain

Synthesize - X

View Synt

View RTL

View Tec

Check Sy

Generate

Implement De

Translate

Map

Place & R

Place

Clock

Async

Pad F

Guide

MPPF

Gene

View

View

Analy

Gene

Gene

Gene

Back

Generate Prog

Programm

Generate

Configure

Update Bitstre

FPGA Design Summary

Design Overview

Summary

IOB Properties

Timing Constraints

Pinout Report

Clock Report

Errors and Warnings

Synthesis Messages

Translation Messages

Map Messages

Place and Route Messa...

Timing Messages

Bitgen Messages

All Current Messages

Detailed Reports

Synthesis Report

Translation Report

Map Report

Place and Route Report

Static Timing Report

Bitgen Report

Synthesis Report

Synthesis Options Summary

HDL Compilation

HDL Analysis

HDL Synthesis

Advanced HDL Synthesis

Low Level Synthesis

Final Report

```
Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block add_g, actual ratio is 0.

=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : add_g.ngr
Top Level Output File Name         : add_g
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 194

Cell Usage :
# BELS                             : 129
#      LUT2                        : 1
#      LUT3                        : 127
#      LUT4                        : 1
# IO Buffers                       : 194
#      IBUF                        : 129
#      OBUF                        : 65
=========================================================================

Device utilization summary:
---------------------------

Selected Device : v1000efg1156-8

 Number of Slices:                   74  out of  12288     0%
 Number of 4 input LUTs:            129  out of  24576     0%
 Number of bonded IOBs:             194  out of    664    29%


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
```

Design Summary | adder

Transcript

```
Started : "Launching Technology Schematic Viewer for add_g.ngc".
```

Console | Errors | Warnings | Find in Files

Snapshot from Xilinx ISE

Synthesis Report – Timing Report

Xilinx - ISE - D:\Gunjan\Xilinx8.2\adder_64\adder_64.ise - [Design Summary]

File Edit View Project Source Process Window Help

Sources

Sources for: Synthesis/I

adder_64

xcv1000e-8fg1156

add_g - beha

Processes

Processes:

User Constrai

Synthesize - X

View Synl

View RTL

View Tec

Check Sy

Generate

Implement De

Translate

Map

Place & R

Place

Clock

Asyn

Pad F

Guide

MPPI

Gene

View

View

Analy

Gene

Gene

Gene

Back

Generate Prog

Programm

Generate

Configure

Update Bitstre

FPGA Design Summary

Design Overview

Summary

IOB Properties

Timing Constraints

Pinout Report

Clock Report

Errors and Warnings

Synthesis Messages

Translation Messages

Map Messages

Place and Route Messa...

Timing Messages

Bitgen Messages

All Current Messages

Detailed Reports

Synthesis Report

Translation Report

Map Report

Place and Route Report

Static Timing Report

Bitgen Report

Synthesis Report

Synthesis Options Summary

HDL Compilation

HDL Analysis

HDL Synthesis

Advanced HDL Synthesis

Low Level Synthesis

Final Report

```
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
No clock signals found in this design

Timing Summary:
---------------
Speed Grade: -8

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 90.059ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 4353 / 65
-------------------------------------------------------------------------
Delay:               90.059ns (Levels of Logic = 66)
  Source:            cin (PAD)
  Destination:       cout (PAD)

  Data Path: cin to cout
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             3   0.744   1.056  cin_IBUF (cin_IBUF)
     LUT3:I0->O            1   0.398   0.736  _n0000_SW0 (N188)
     LUT3:I2->O            2   0.398   0.920  _n0000 (_n0000)
     LUT3:I2->O            2   0.398   0.920  _n02251 (_n0225)
     LUT3:I2->O            2   0.398   0.920  _n00011 (_n0001)
     LUT3:I2->O            2   0.398   0.920  _n02371 (_n0237)
     LUT3:I2->O            2   0.398   0.920  _n00021 (_n0002)
     LUT3:I2->O            2   0.398   0.920  _n02291 (_n0229)
     LUT3:I2->O            2   0.398   0.920  _n00031 (_n0003)
     LUT3:I2->O            2   0.398   0.920  _n02211 (_n0221)
     LUT3:I2->O            2   0.398   0.920  _n00041 (_n0004)
```

Design Summary | adder

Transcript

```
Started : "Launching Technology Schematic Viewer for add_g.ngc".
```

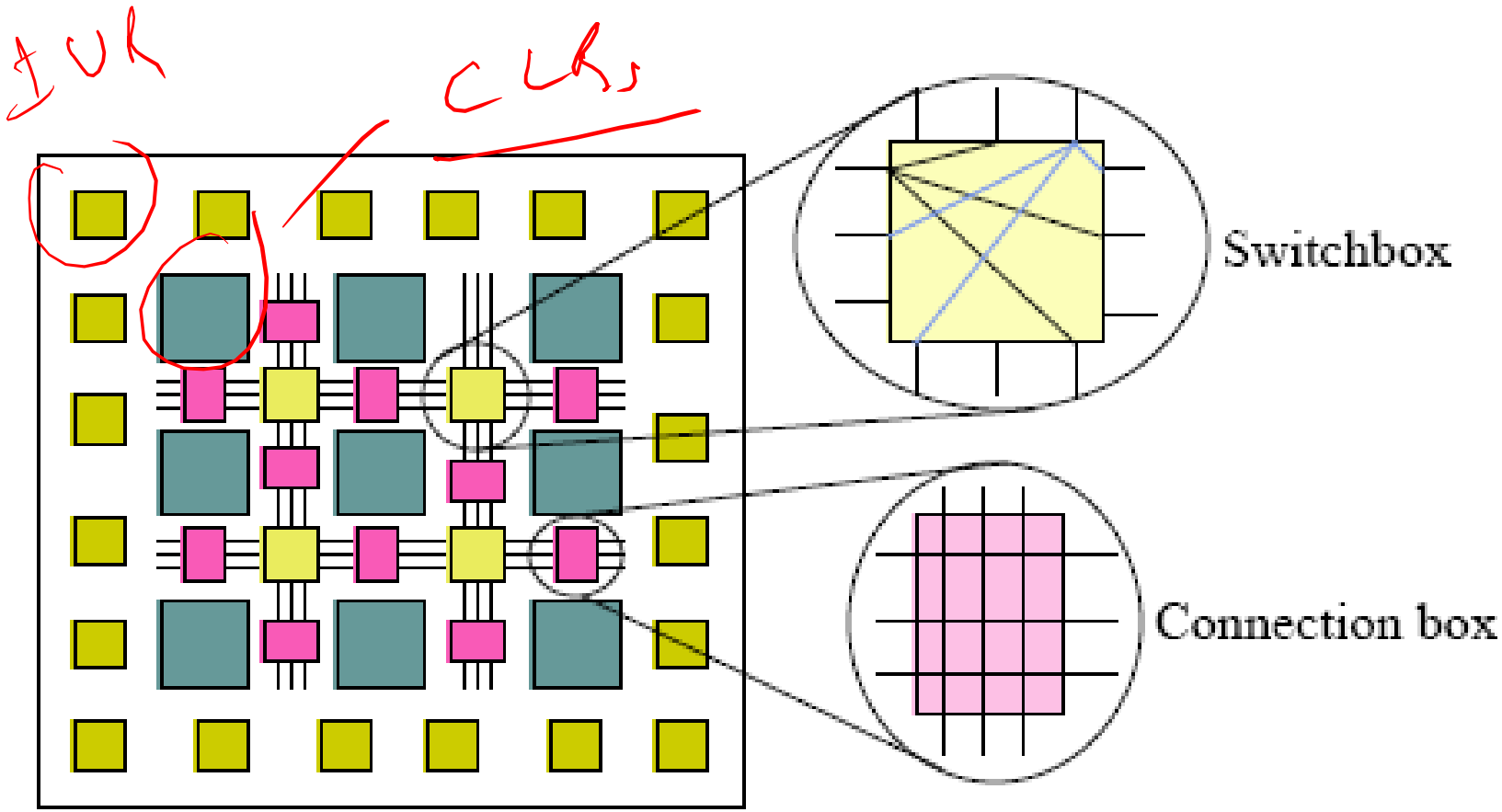
Console | Errors | Warnings | Find in Files

Snapshot from Xilinx ISE

Lecture 34: FPGA Architecture

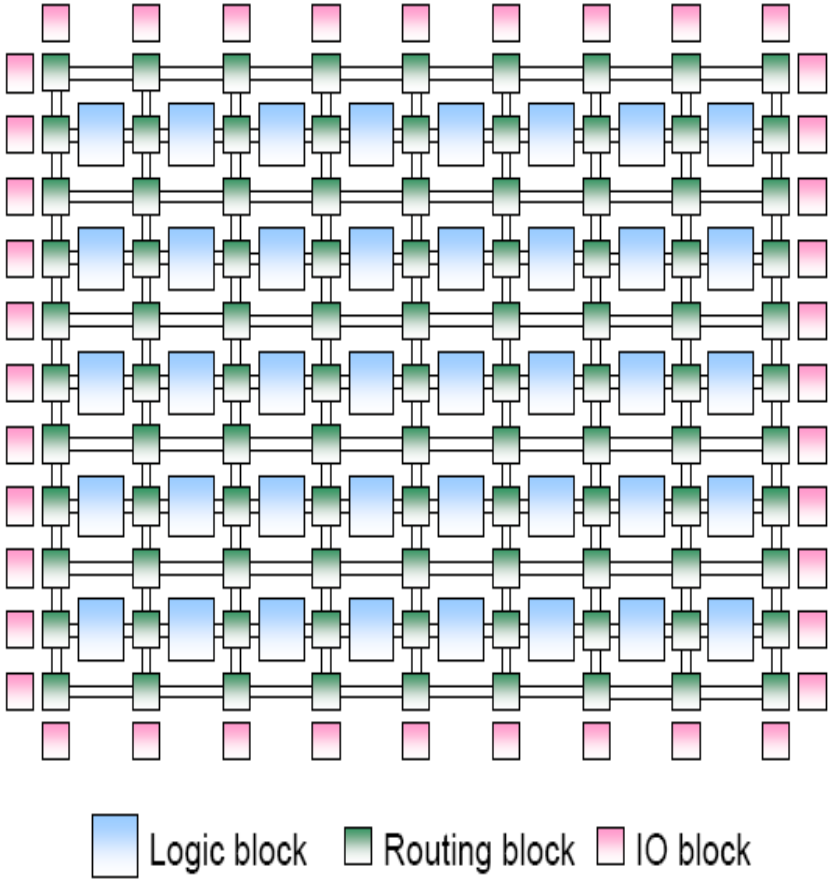
M. Balakrishnan

Generic 2D FPGA

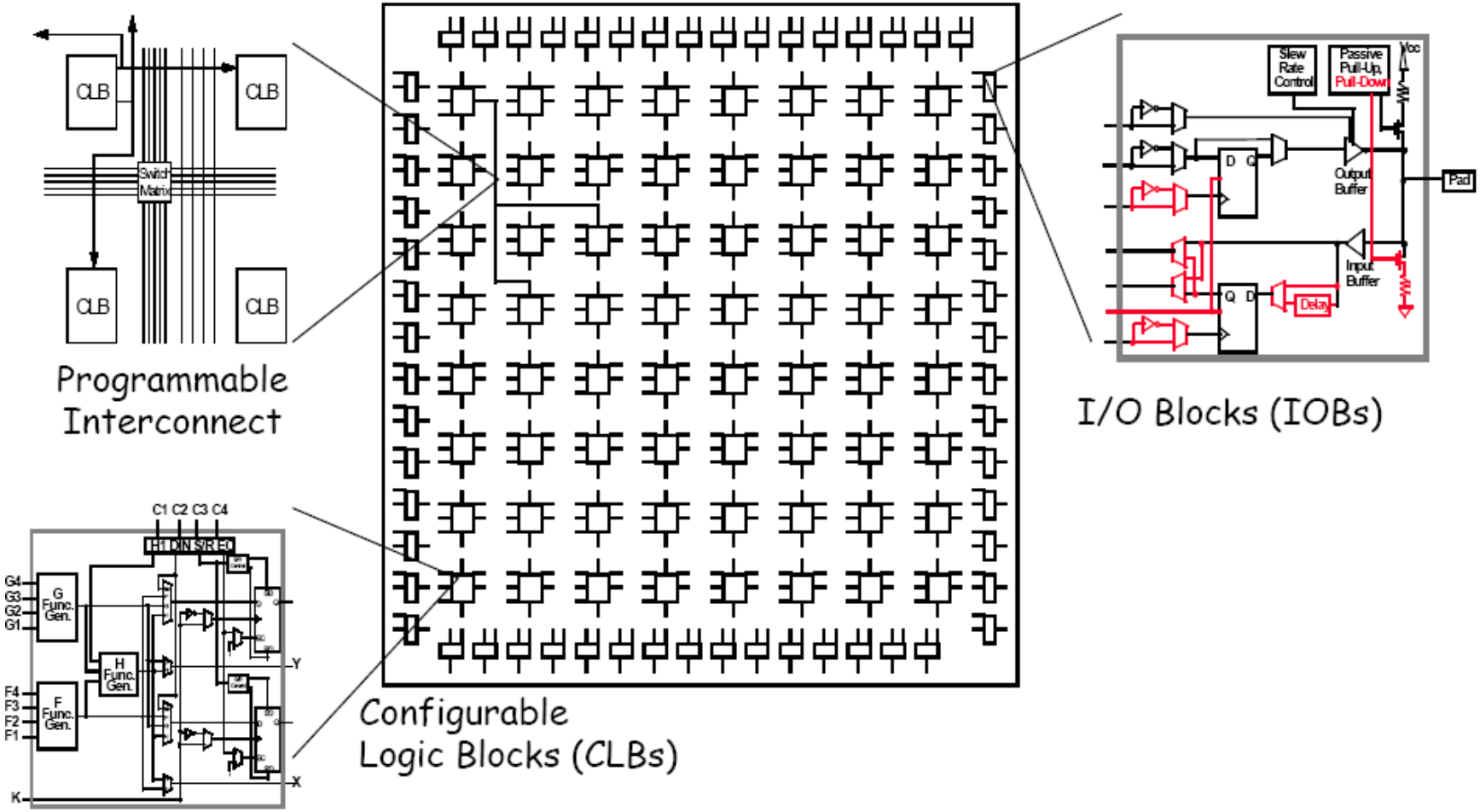


FPGA Architecture

- An FPGA is an array of programmable logic elements that can be connected to inputs or outputs.

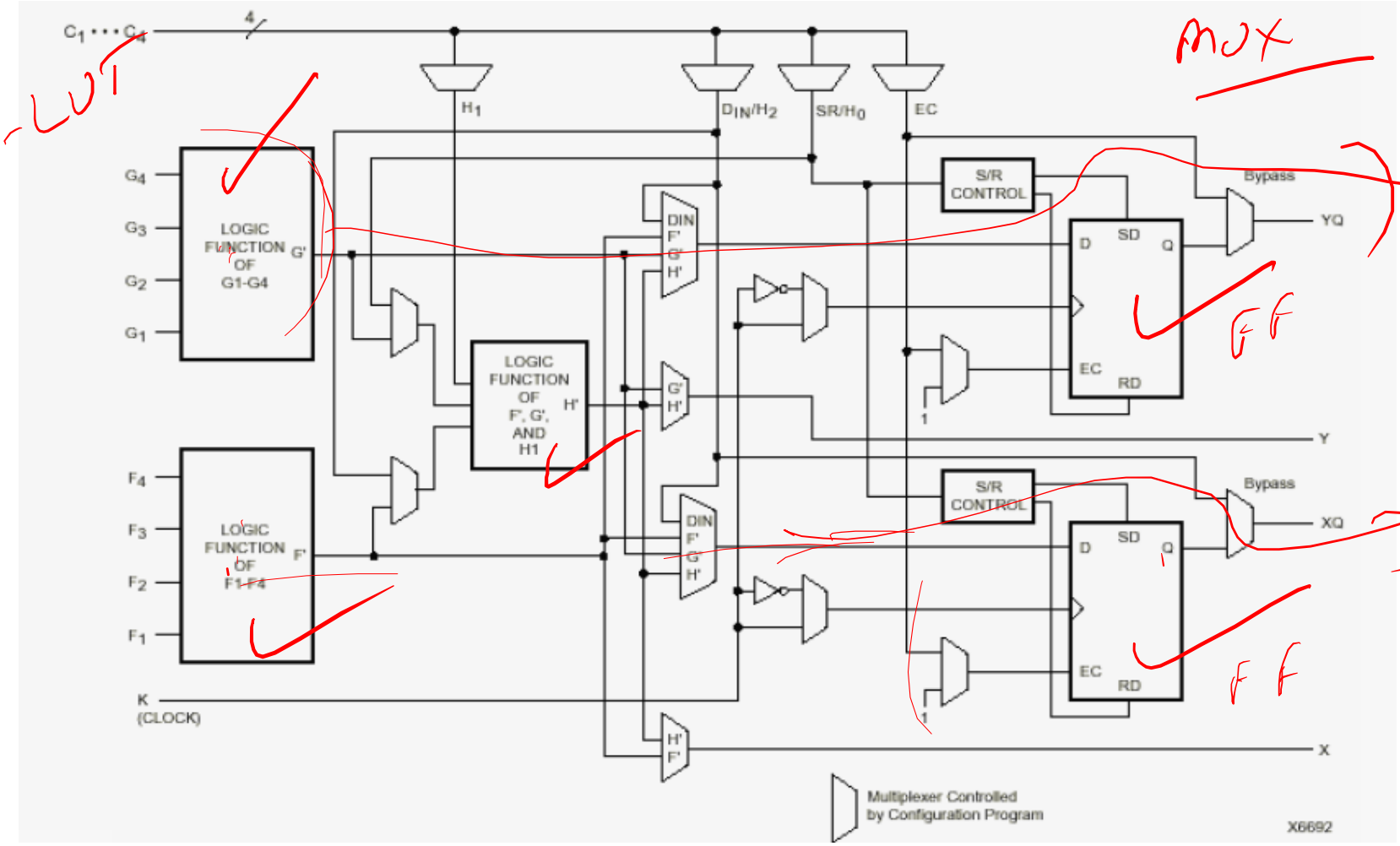


SRAM Based FPGA - XILINX



Xilinx 4000 CLB

Configurable

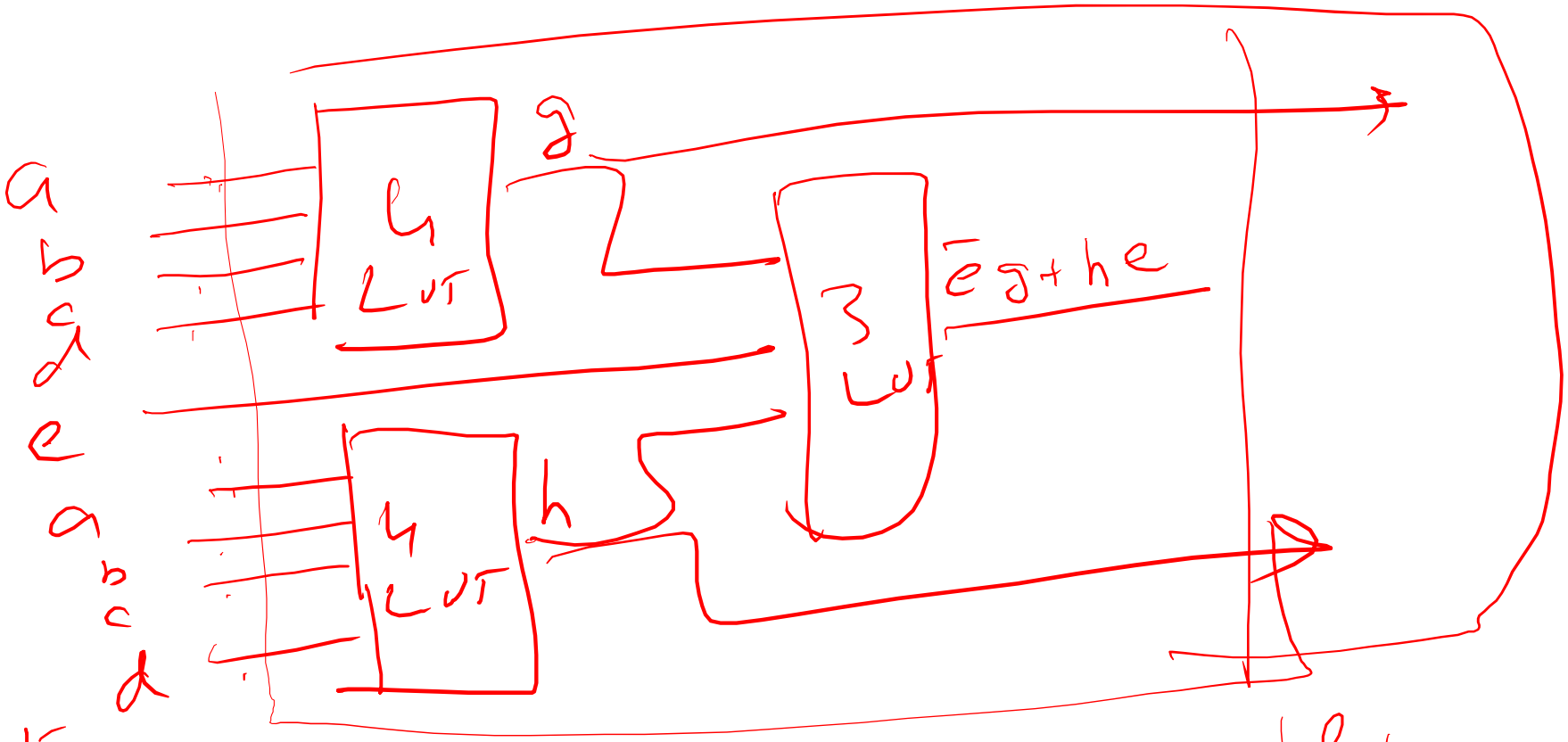


4-LUT

MUX

FF

FF



9 inputs

$f(a, b, c, d)$

5 - variables

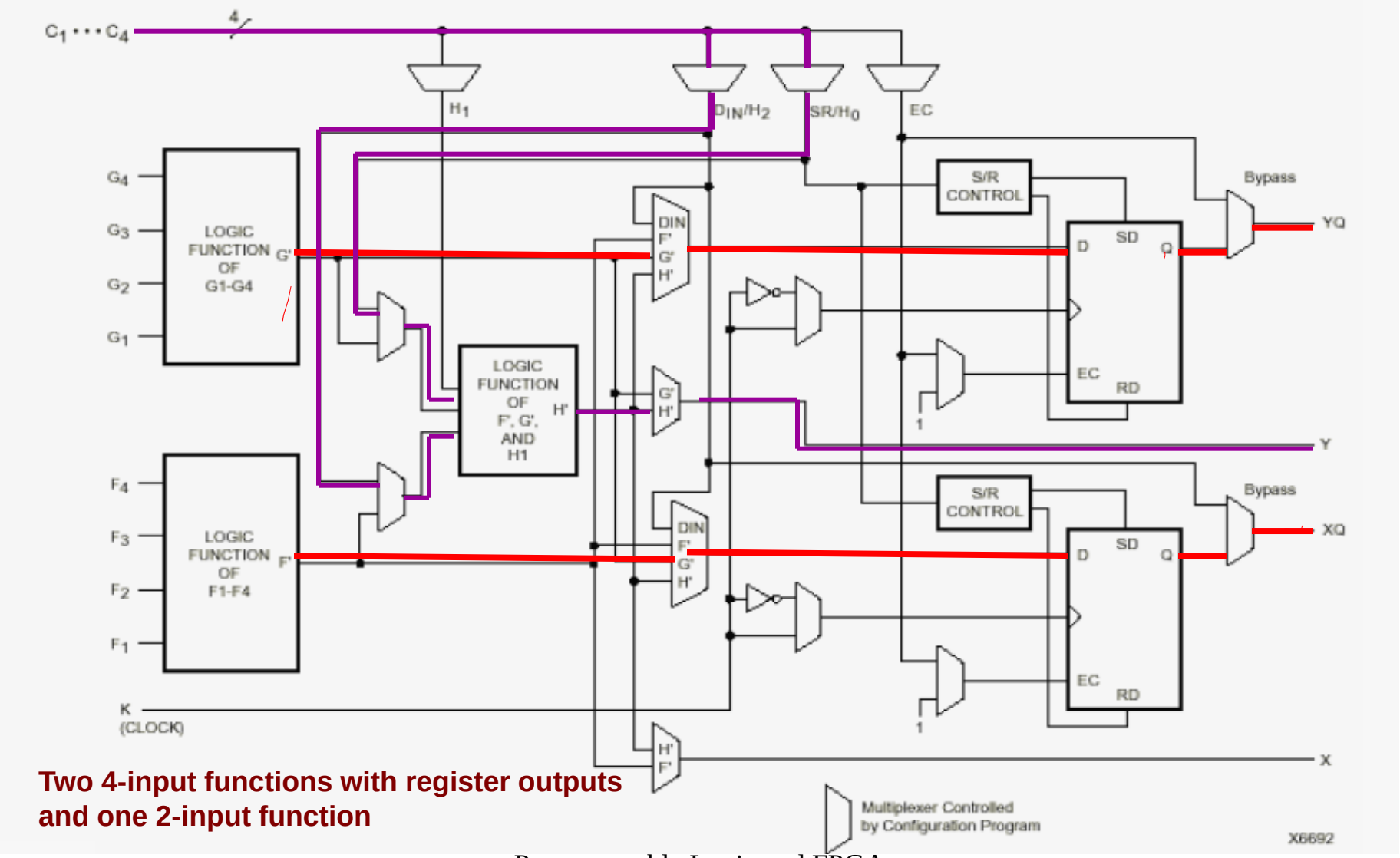
NU

9 - variables

Answer

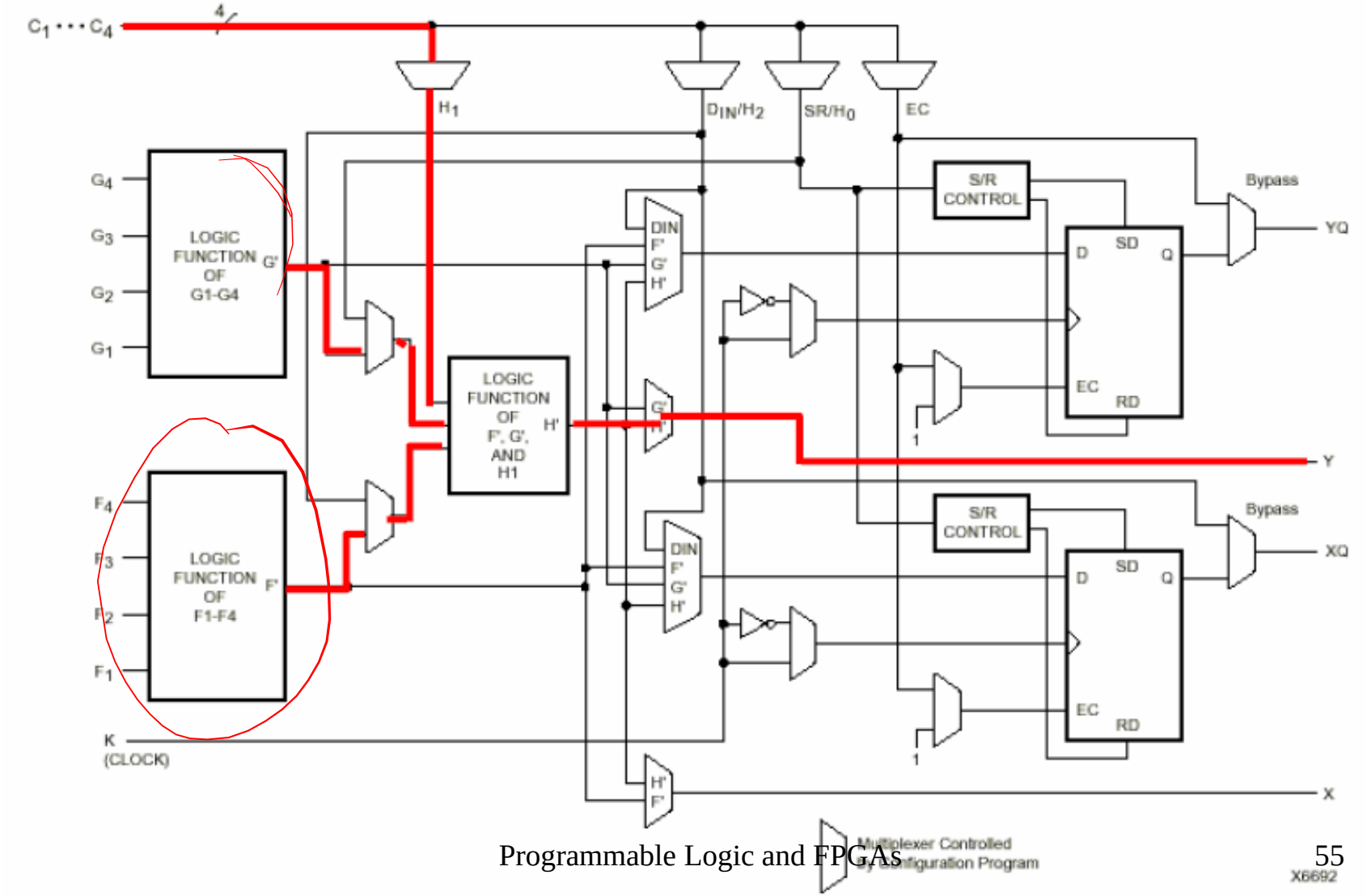
$f(a,b,c,d,e) = g(a,b,c,d)\bar{e} + h(a,b,c,d)e$

Implementing Combinational Logic



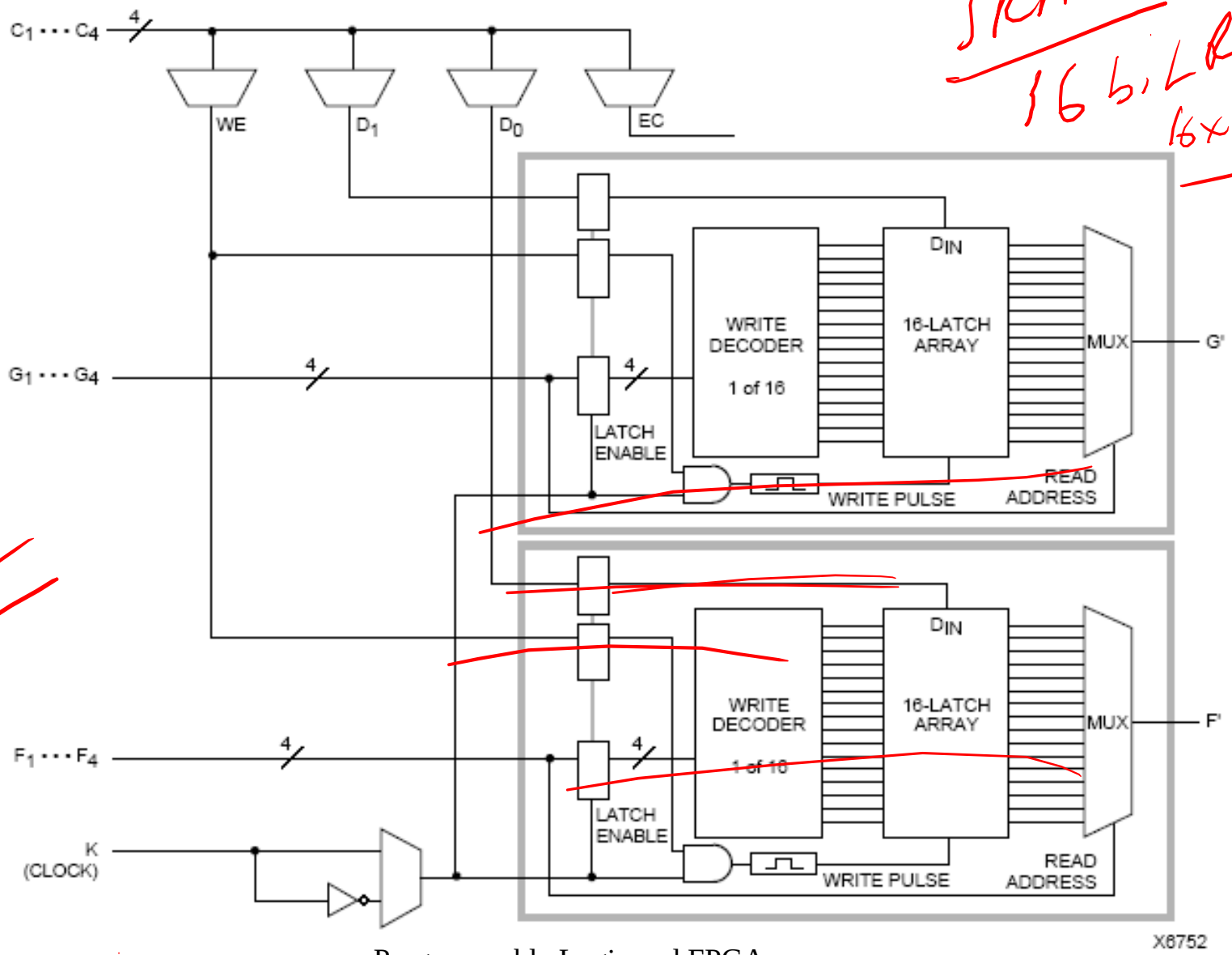
Two 4-input functions with register outputs and one 2-input function

5 input Function



Single Port RAM

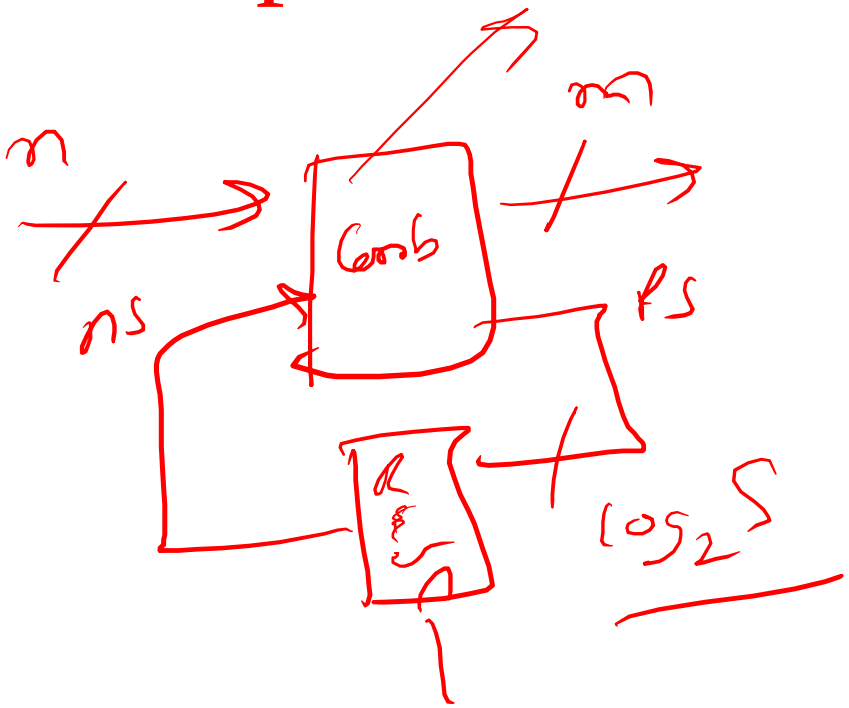
SRAM
16 b. LRAM
16x2



16x2

http://www.xilinx.com/bvdocs/publications/4000.pdf

Function Decomposition

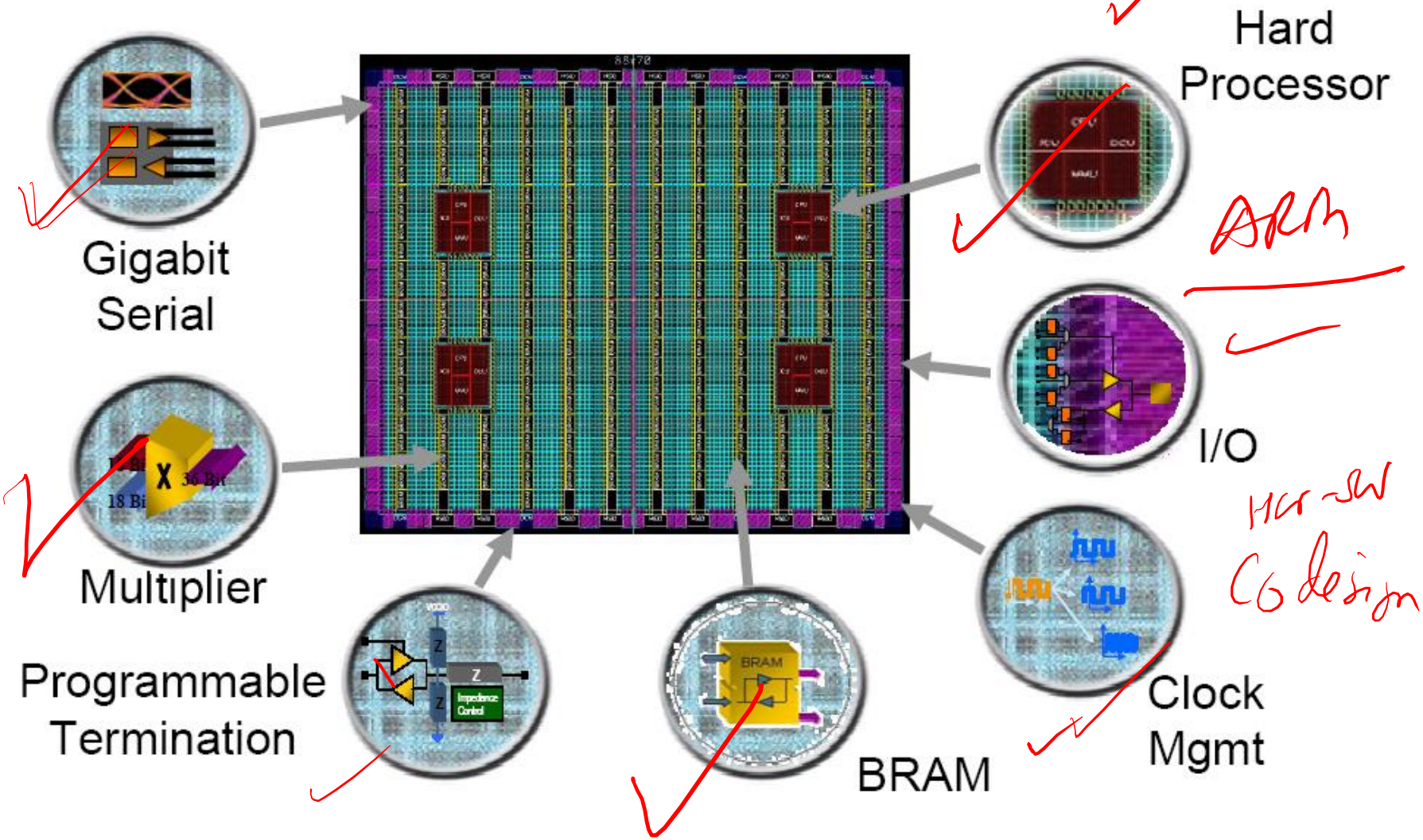


8 5k bits

4 CLBs

XC4000

Platform Computing



Courtesy of David B. Parlour, ISSCC 2004 Tutorial,
"The Reality and Promise of Reconfigurable Computing in Digital Signal Processing"

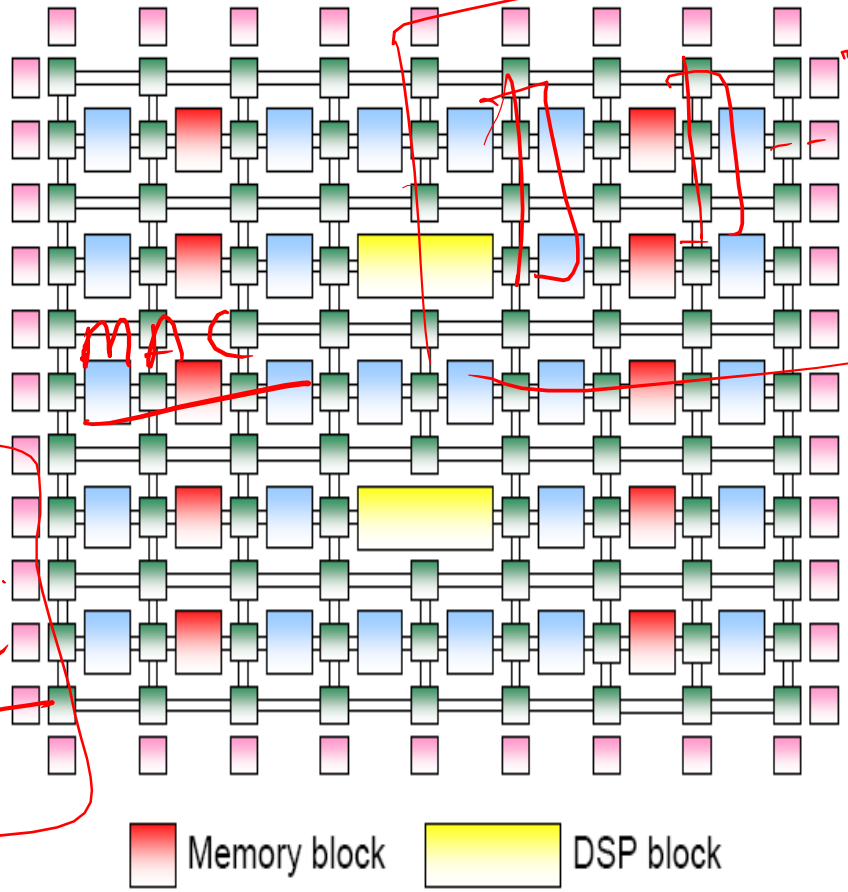
$x_i \sin\theta + y_i \cos\theta$

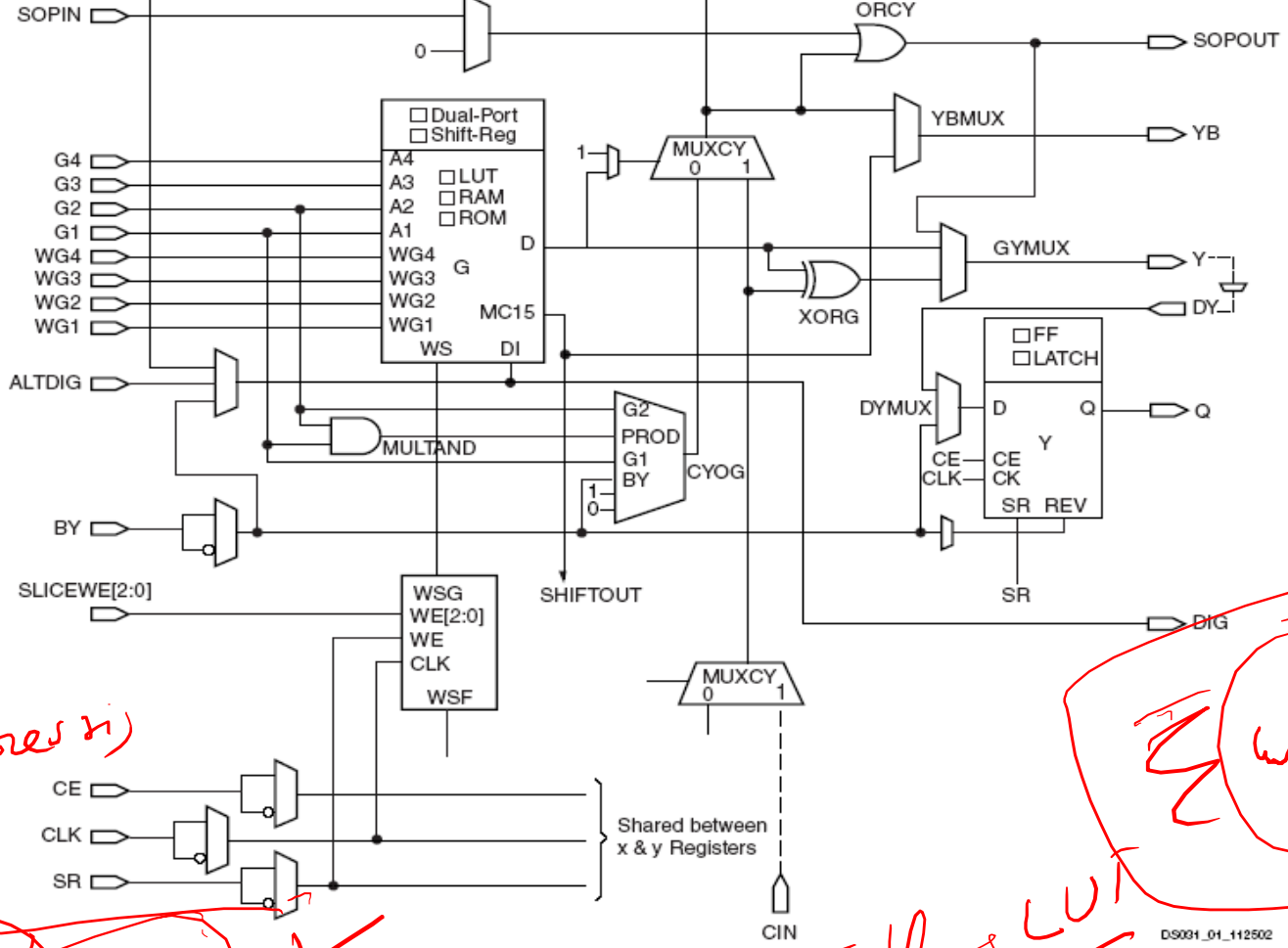
Modern FPGA Fabric

Programmable fabric

- Memory blocks
- DSP blocks

$$\sum_{i=1}^{n} w_i x_i$$





Signal Processing

$\sum w_i x_i$

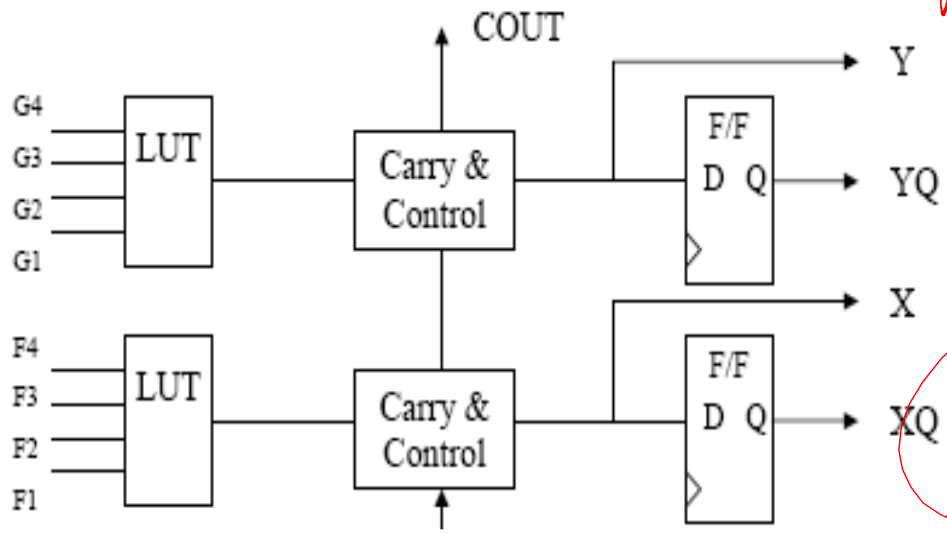
LUT
RAM
ROM

Truth Table → LUT

Configuration bit-file

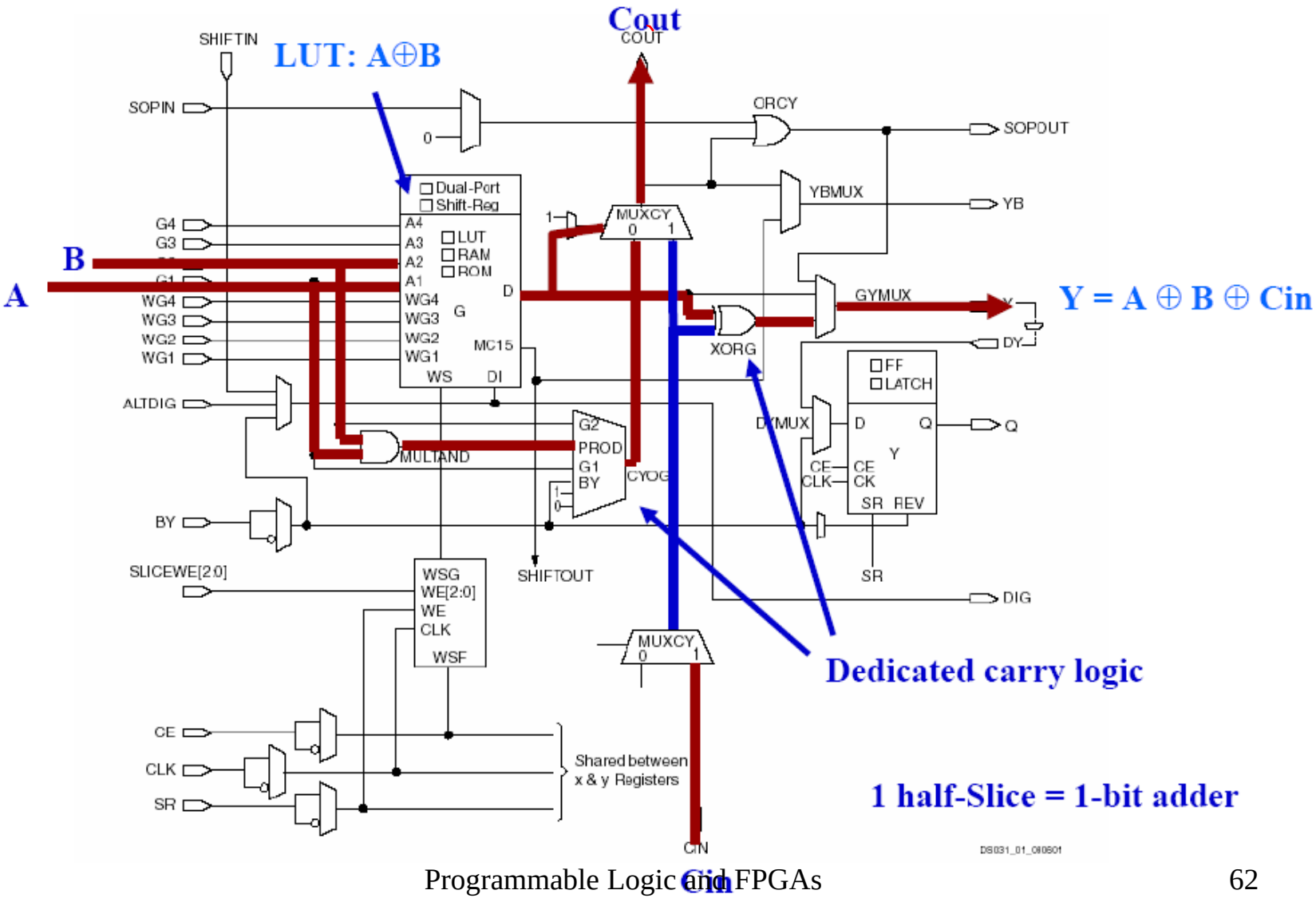
FPGA Performance Issues

- Flexible logic – unused logic and additional delays
- Interconnects through switches – additional delays

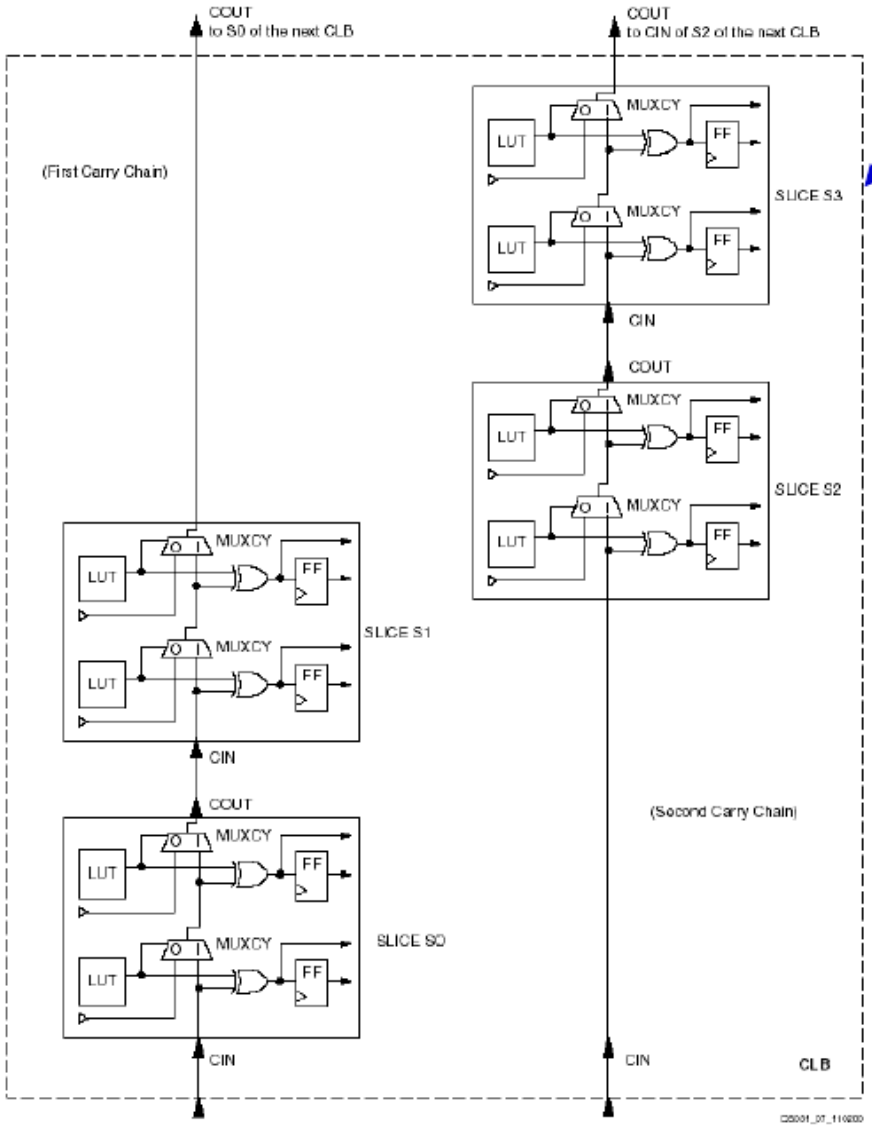


Source: xilinx.com

Adder Implementation

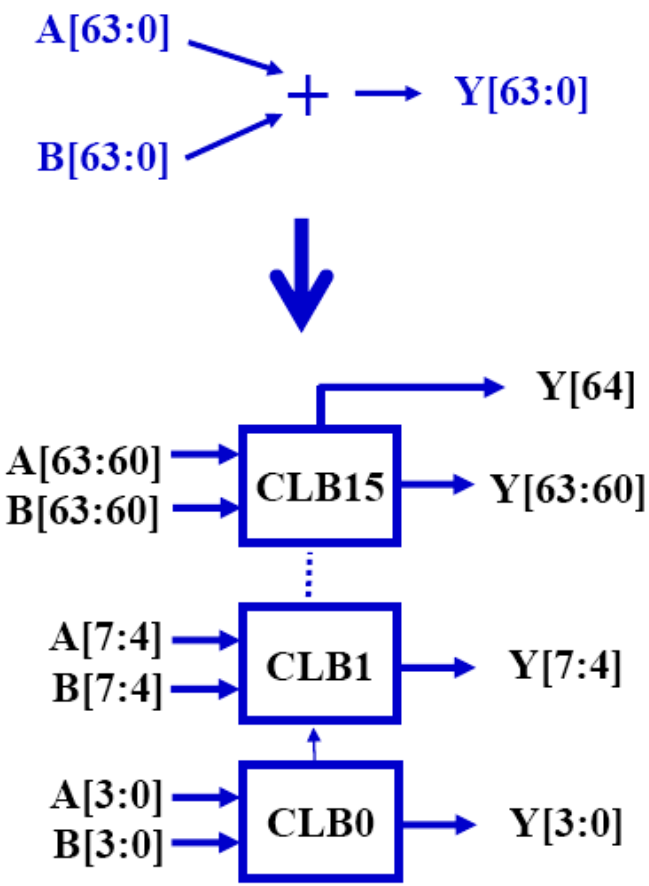


Carry Chain Implementation



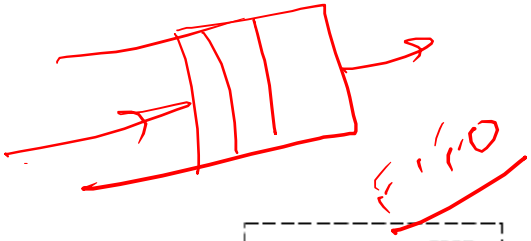
1 CLB = 4 Slices = 2, 4-bit adders

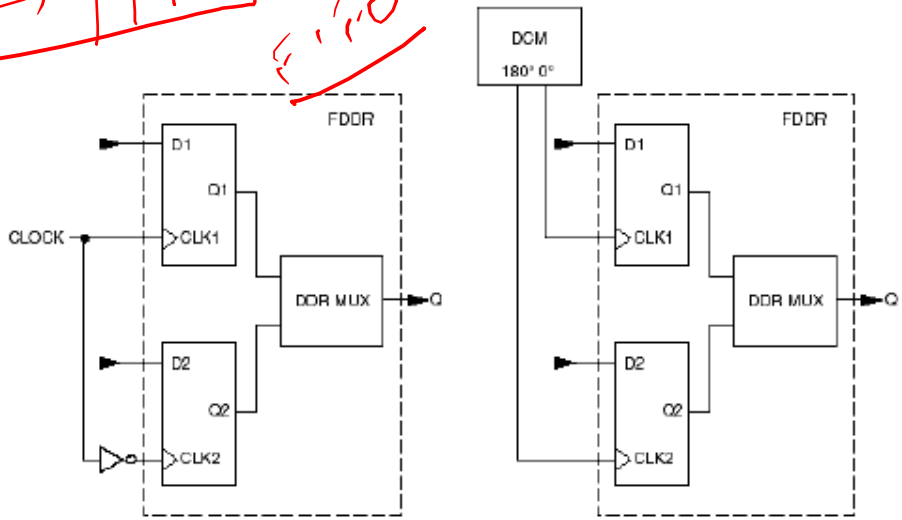
64-bit Adder: 16 CLBs



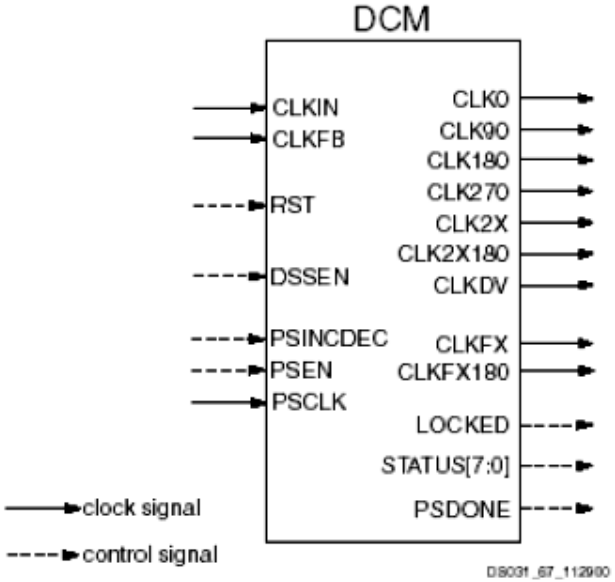
CLBs must be in same column

Other Features

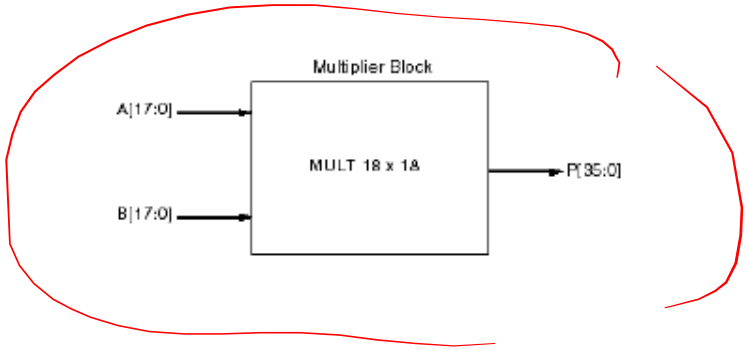




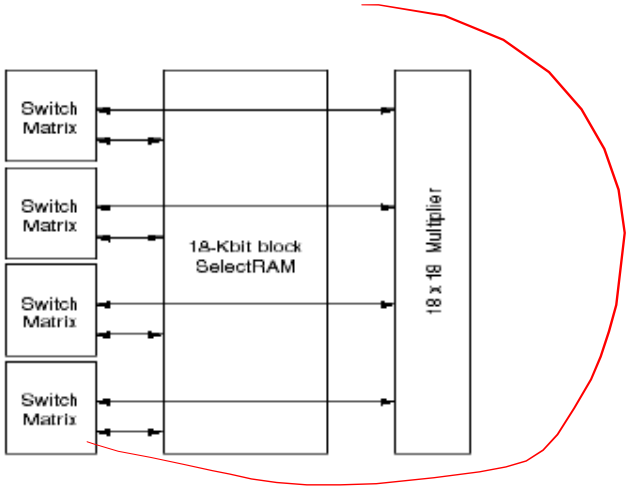
Double Data Rate registers



Digital Clock Manager



Embedded Multiplier



Block SelectRAM

Technology independent

Lecture 35: FPGA Technology Mapping

M. Balakrishnan

Design

VHDL

Technology

FPGAs

IP modules

CLB

Definition

Technology mapping is also referred to as *library binding*.

Given a Boolean network and a characterized cell library, generate a mapping of the network components onto cell library components with the objective of cost optimization or delay optimization.

ASIC

2 ns
5 ohms

1.5 ns
66 ohms

Input & Library

- Input: Boolean network - Technology independent optimized network; typically a multi-level network
- Library:
  - Characterization in terms of area, delay and power
  - Enumerated or implicit library cells

Typical Library

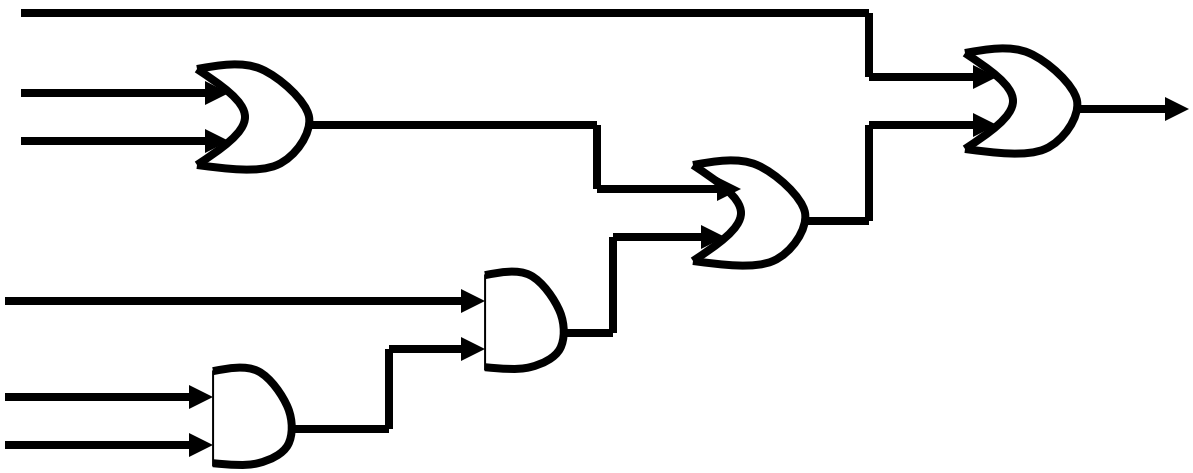
A typical simple library cell :-

- a single output combinational logic function
- cost in terms of area
- delay in terms of propagation delays for each input/output pair and as a function of load and/or fanout. Sometimes only the worst case values are stored.
- power in terms of average current

Network Covering

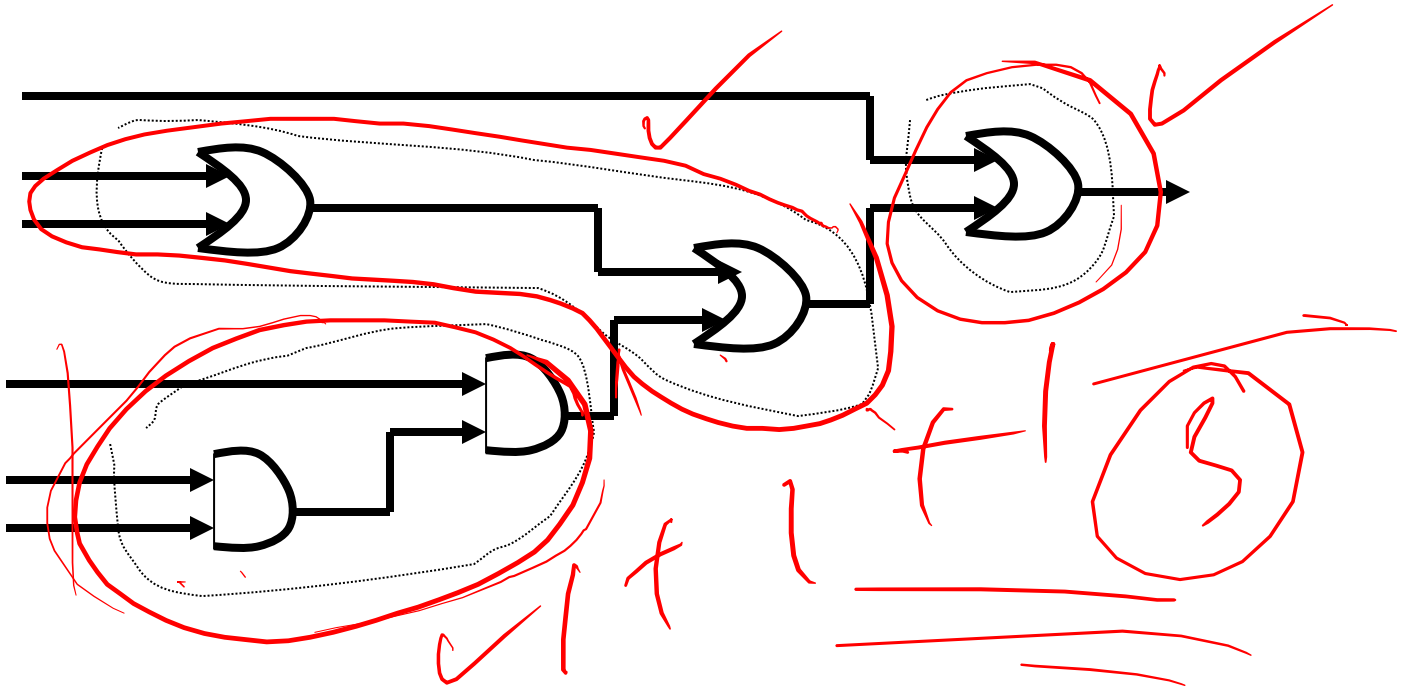
Network covering implies replacement of the sub-networks of the original network with cell library instances. Covering entails recognizing the equivalence of library cell to the identified sub-network and selecting adequate number of them to cover the whole network.

Example 1



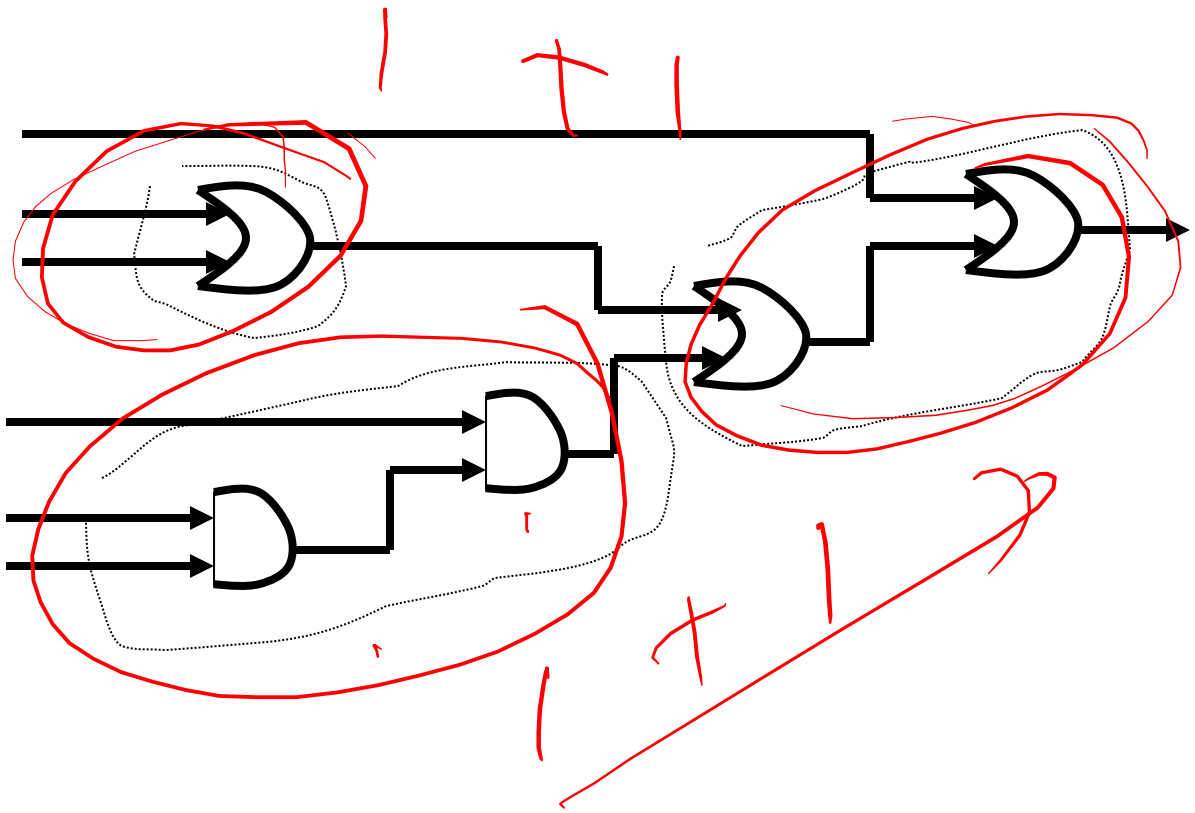
Cell library consists of two and three input gates

Example: First Mapping



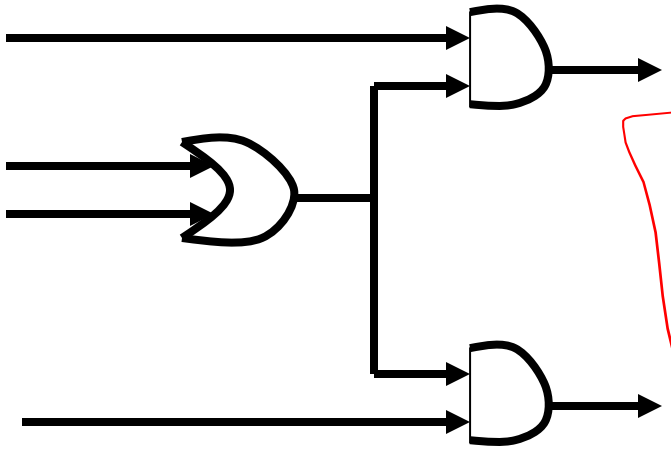
Cell library consists of two and three input gates

Example: Second Mapping



Cell library consists of two and three input gates

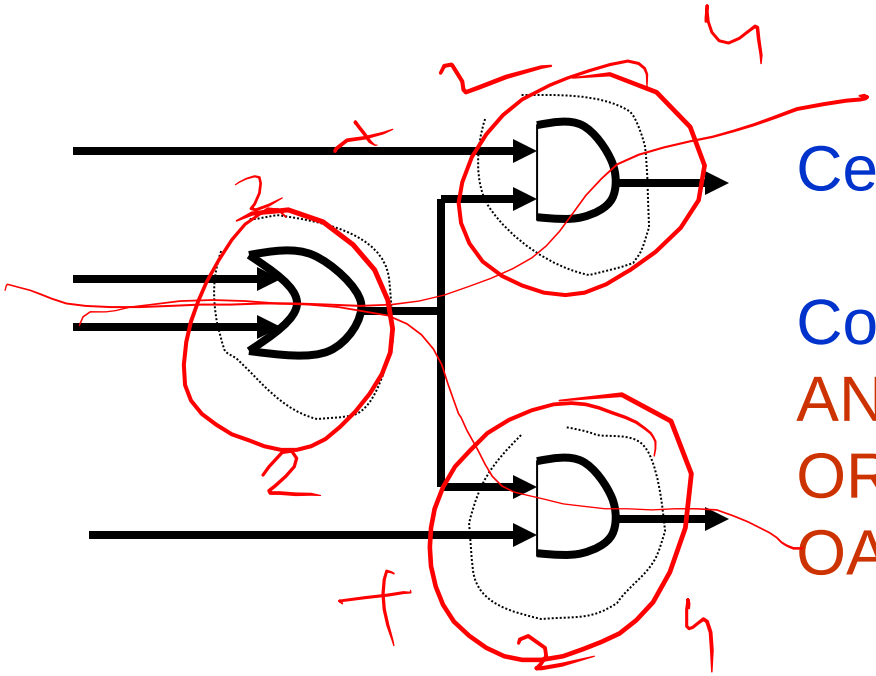
Example 2



Cell library consists of

Component	Area	Delay
AND2	3	2
OR2	3	2
OA21	5	3

Example: First Mapping

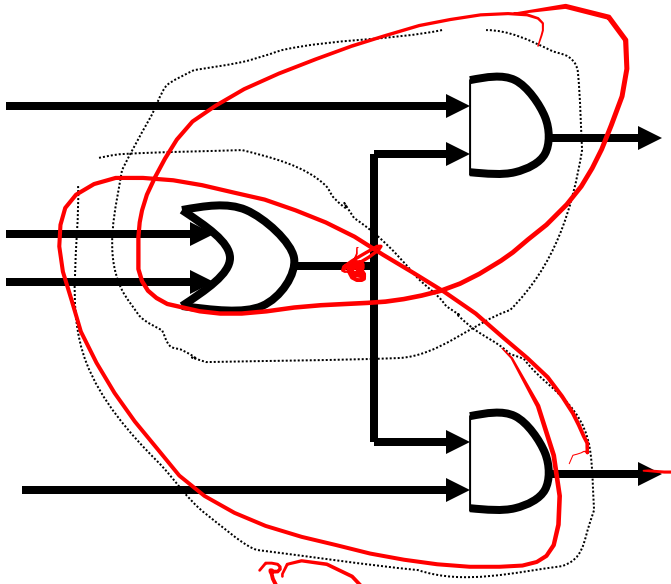


Cell library consists of

Component	Area	Delay
AND2	3	2
OR2	3	2
OA21	5	3

Area = 9, Delay = 4

Example: Second Mapping

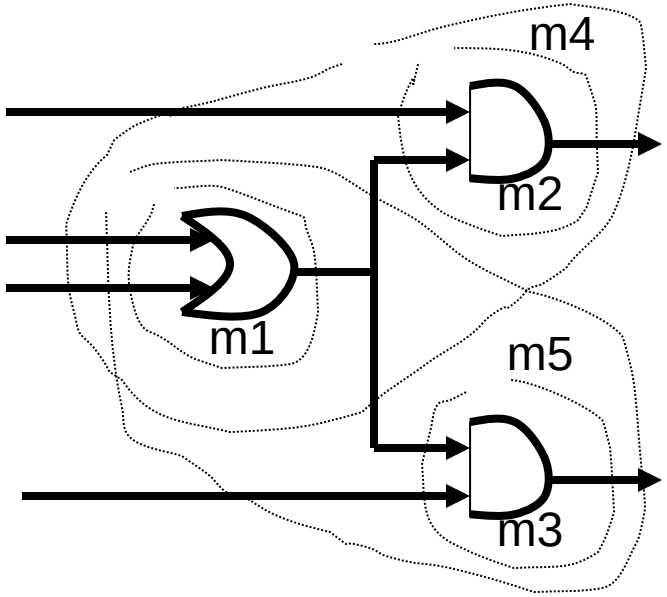


Cell library consists of

Component	Area	Delay
AND2	3	2
OR2	3	2
OA21	5	3

Area = 10, Delay = 3

Example



Cell library consists of

Component	Area	Delay
AND2	3	2
OR2	3	2
OA21	5	3

$$(m1 + m4 + m5)(m2 + m4)(m3 + m5)(m2' + m1)(m3' + m1) = 1$$

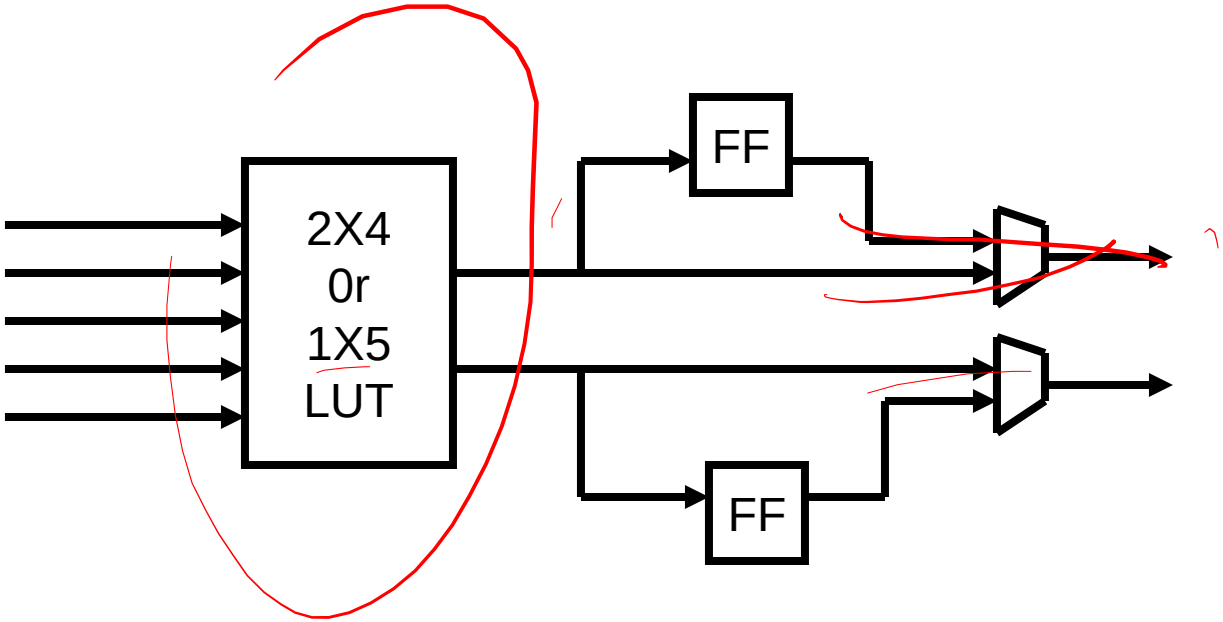
FPGA Structures & Mapping

LUT Based FPGAs

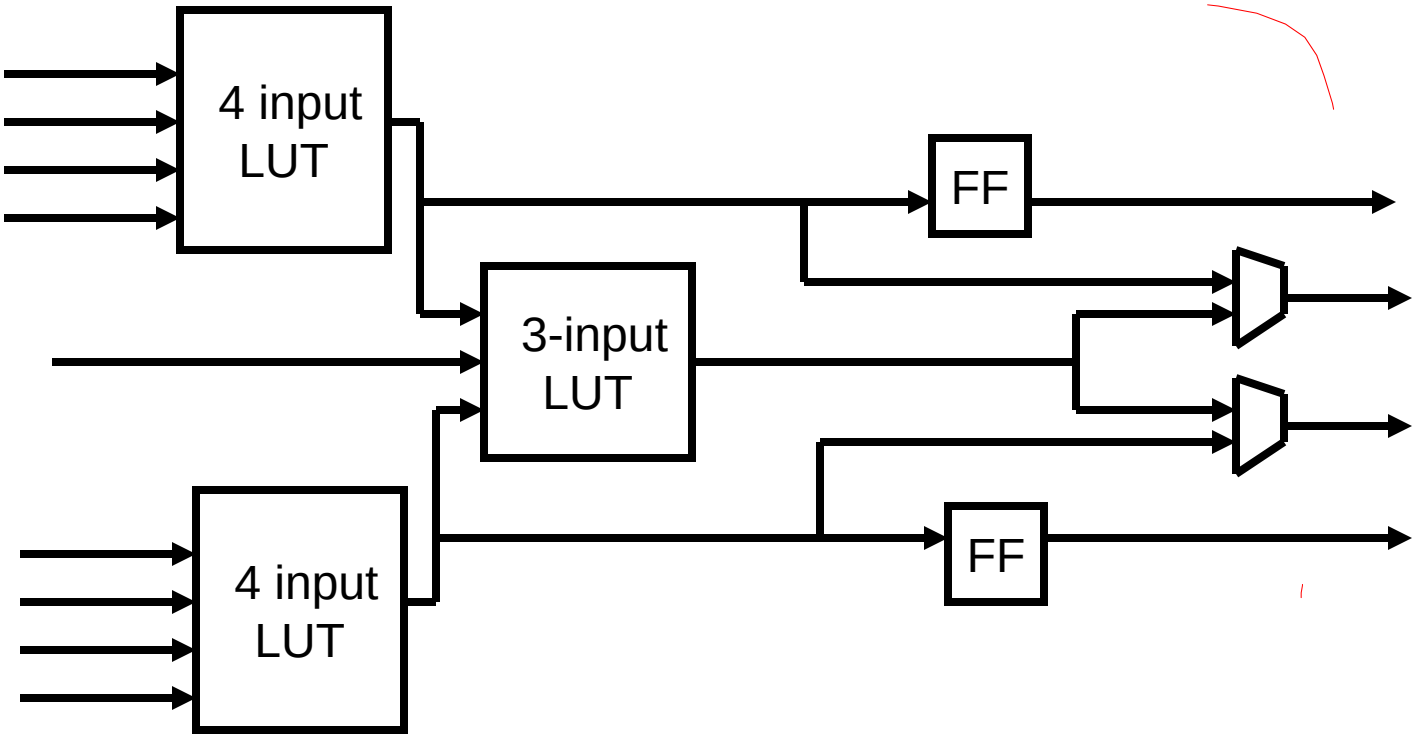
In LUT based FPGAs (example XILINX FPGAs) the building blocks are LUTs and Flip-Flops. A n-input LUT can implement all functions of n-variables.

The FPGA itself is composed of CLB's with each CLB containing multiple LUT's and flip-flops which makes the technology mapping problem more complex.

XC3000 CLB



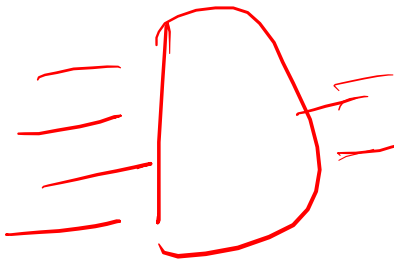
XC4000 CLB



Mapping Objectives

- Cost optimal mapping
  - Minimizing the number of LUTs
  - Minimizing the number of CLBs
- Delay optimal mapping
  - Minimizing the number of LUT levels
  - Minimizing the delays (including routing delays)

Cost Optimal Mapping

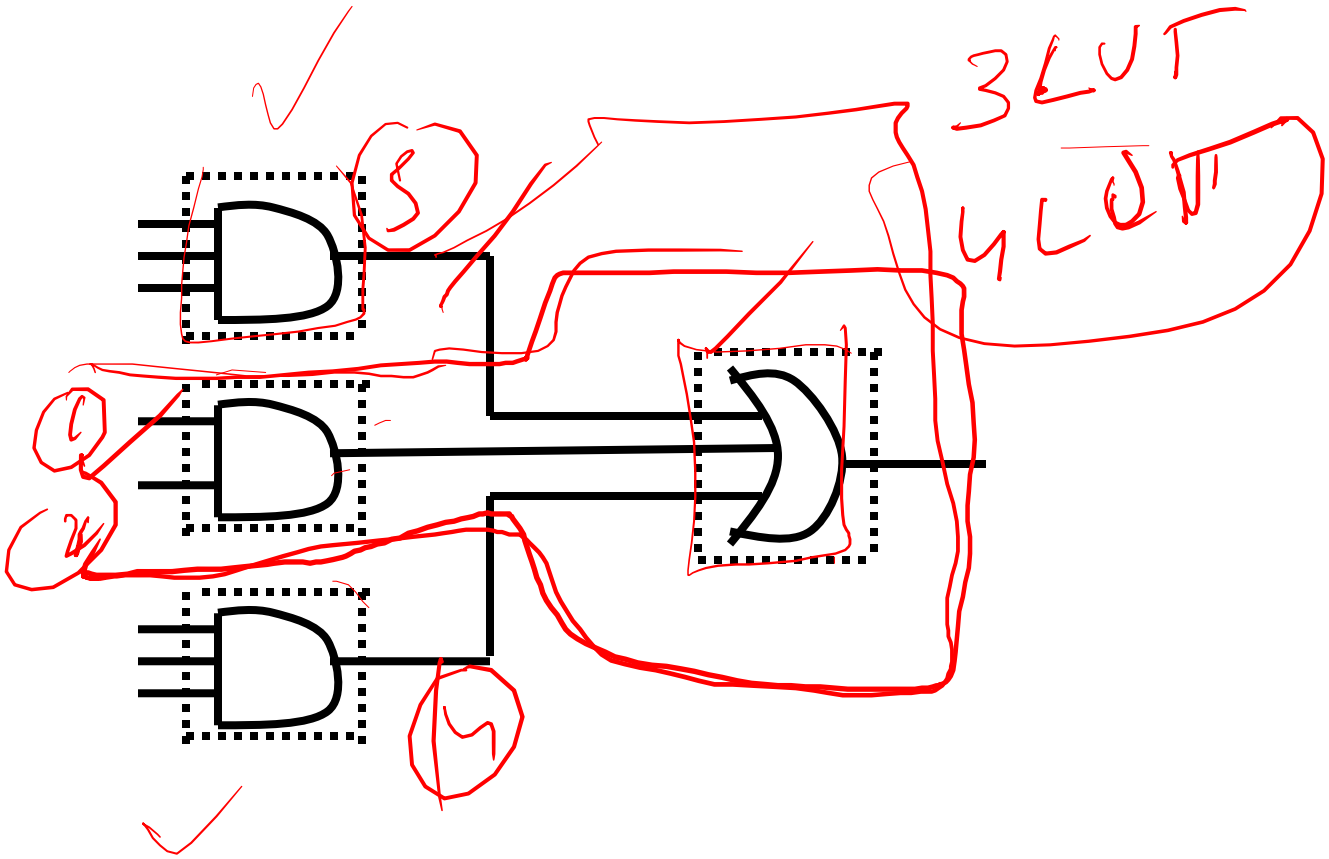


The problem of k-input LUT maps can be mapped to the problem of bin packing. We have to minimize the number of bins each with a capacity of k.

Assume the starting point is a gate-level netlist with each gate containing less than equal to k inputs.

Each gate can be packed into one bin.

Example: Simple Mapping

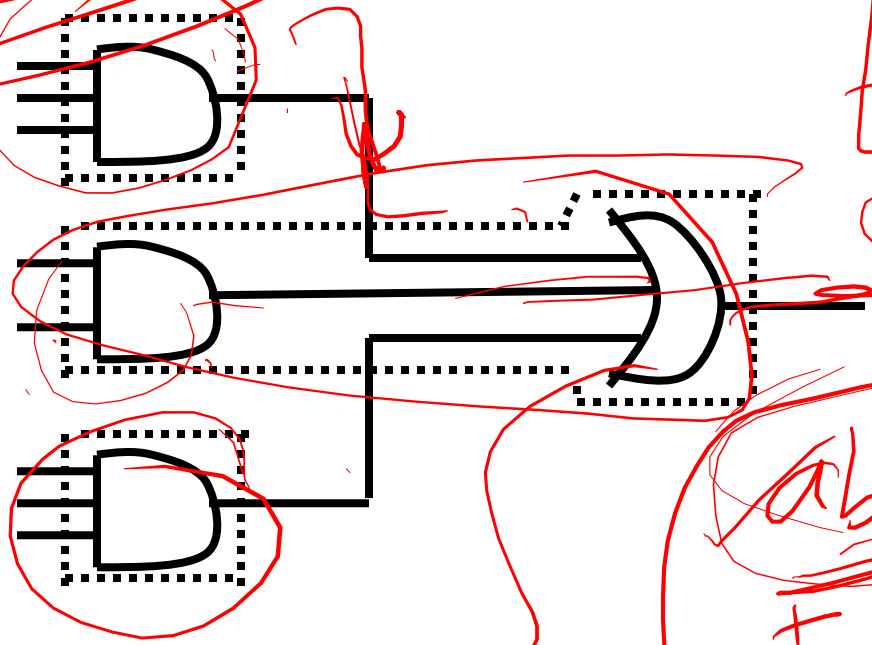


$a.b.\bar{c}$ $c\bar{d}e$

Sum of Products: Bin Packing

- Select the product term with the most number of variables and fit it into any table where it fits and if it doesn't fit anywhere add a new table
- The table with the fewest number of unused inputs is declared as final
- Associate this output with the first table that can accept it

Example: 4-input LUT



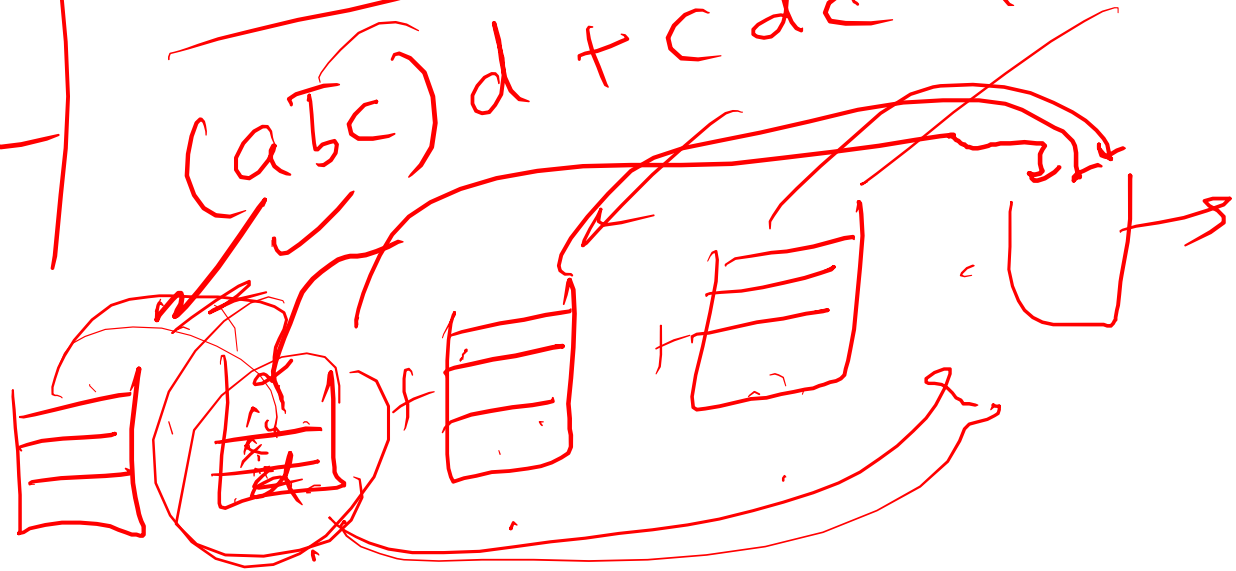
2 LUT

NAND
NOR
EXOR
EXNOR

$$a\bar{b}cd + c\bar{d}e + b\bar{d}e$$

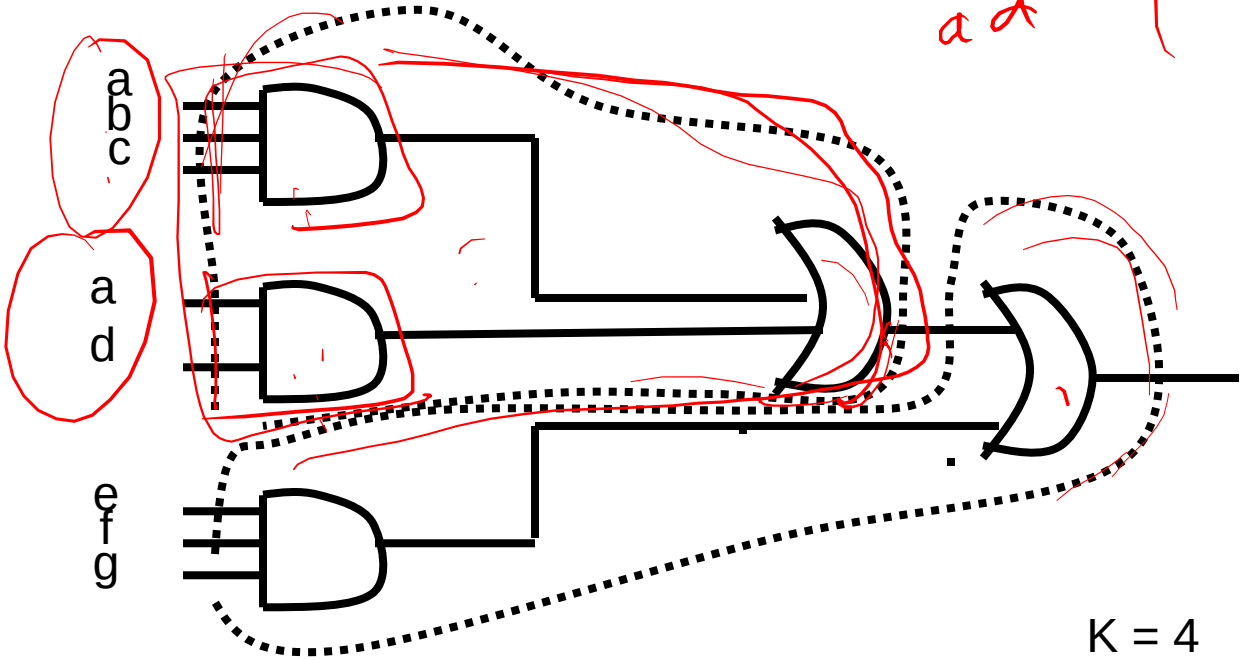
$$(a\bar{b}c)d + c\bar{d}e + b\bar{d}e$$

3 LUT +



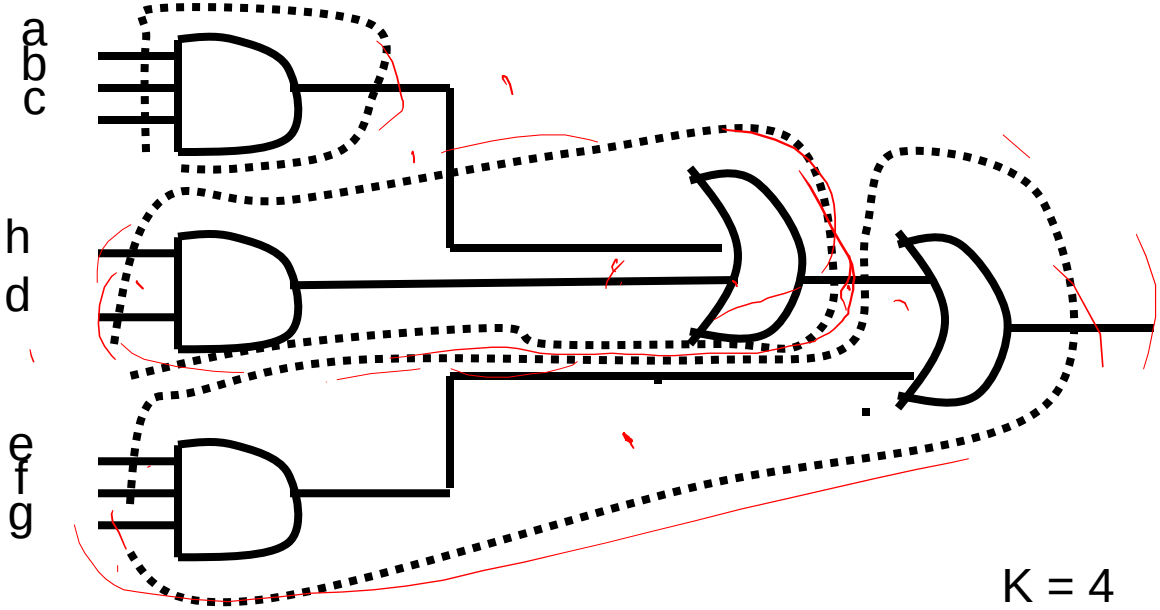
Example: Overlapping Inputs

abc
ad
(a, b, c, d)
= 4

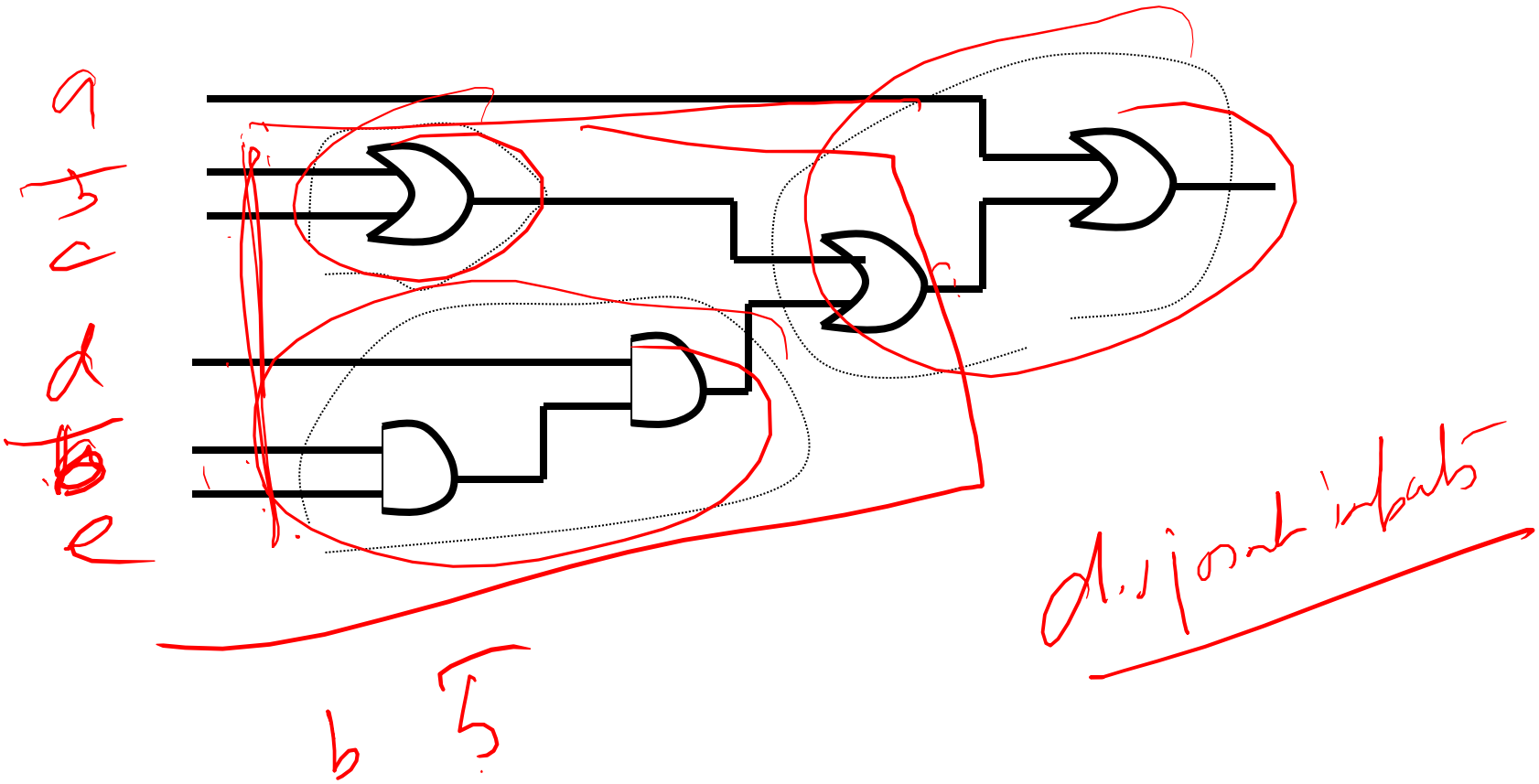


K = 4

Example: Decomposition



Example: 3 input LUT



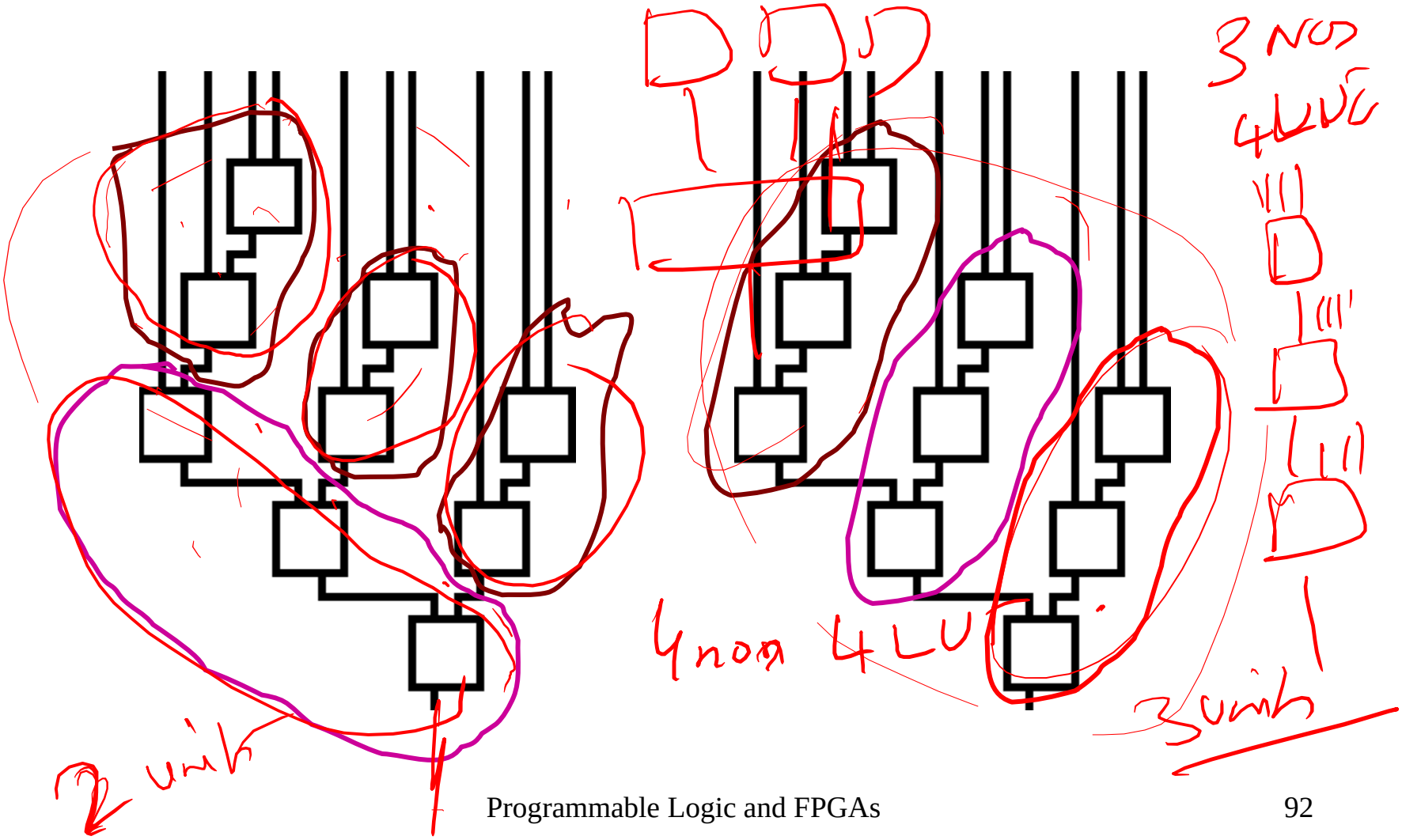
FPGA Technology Mapping: Issues

LUT Mapping

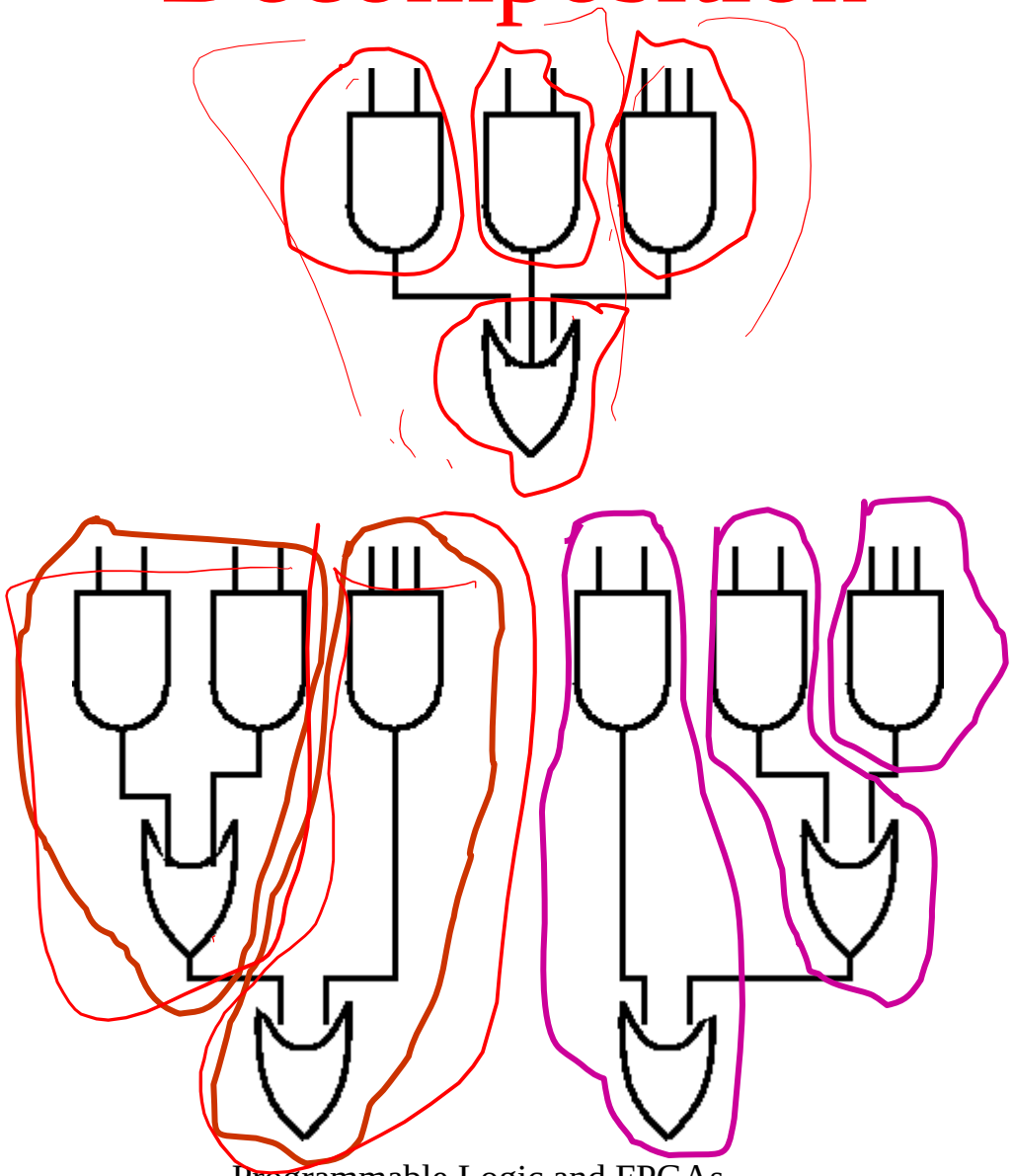
Starting from a technology independent optimized circuit, produce a minimal LUT cover for the circuit. The complexities are due to the following reasons.

- Fanout nodes
- Reconvergence
- Node decomposition and packing

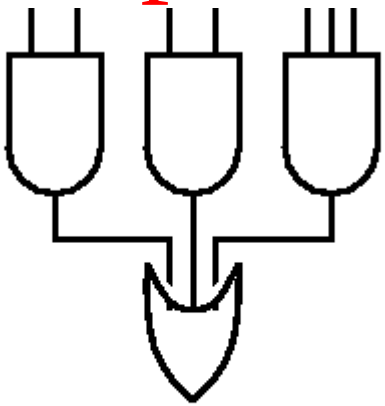
Area vs. Delay

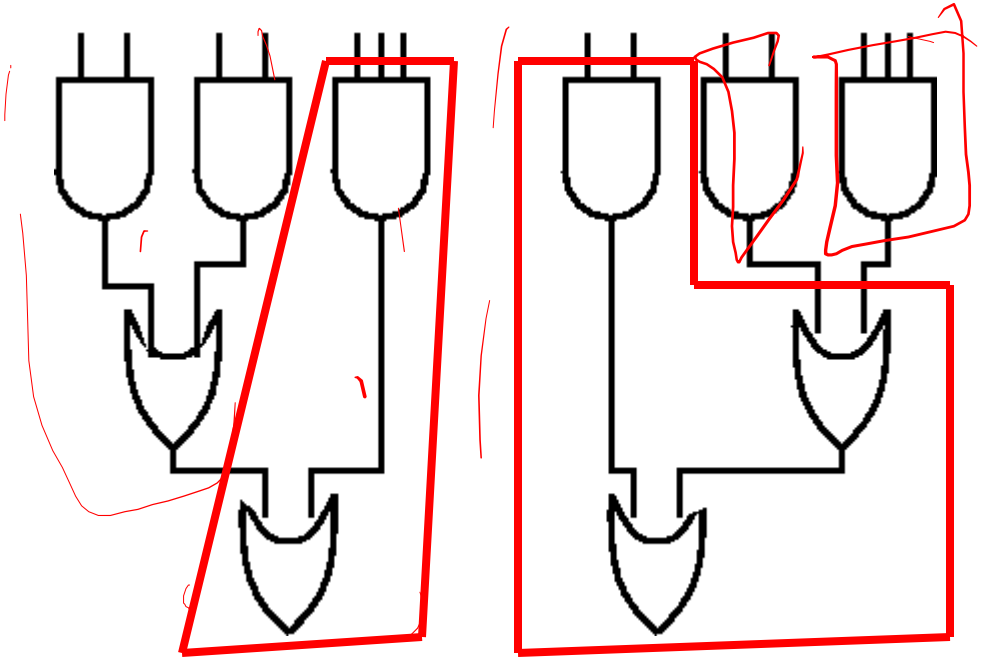


Decomposition

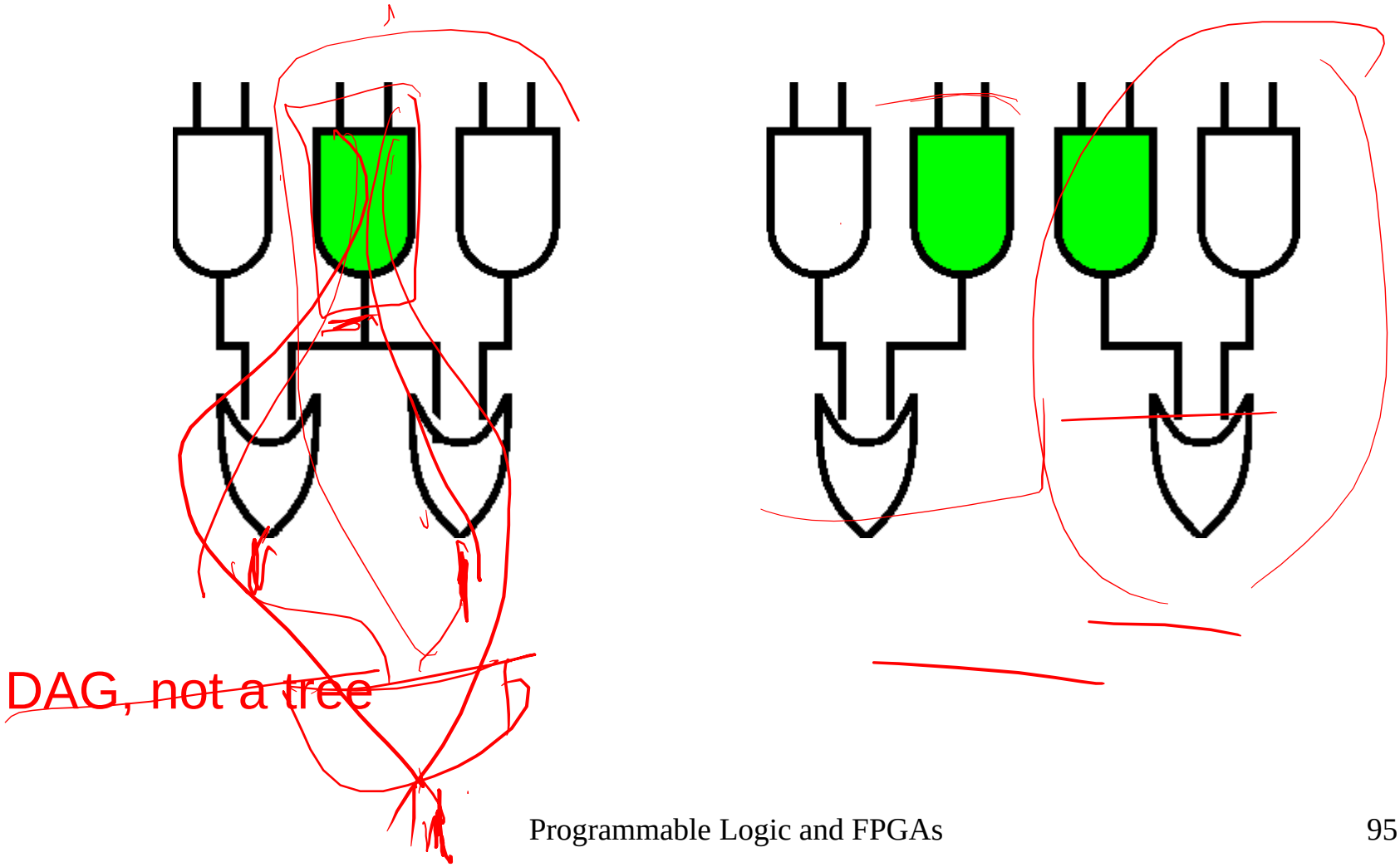


Decomposition

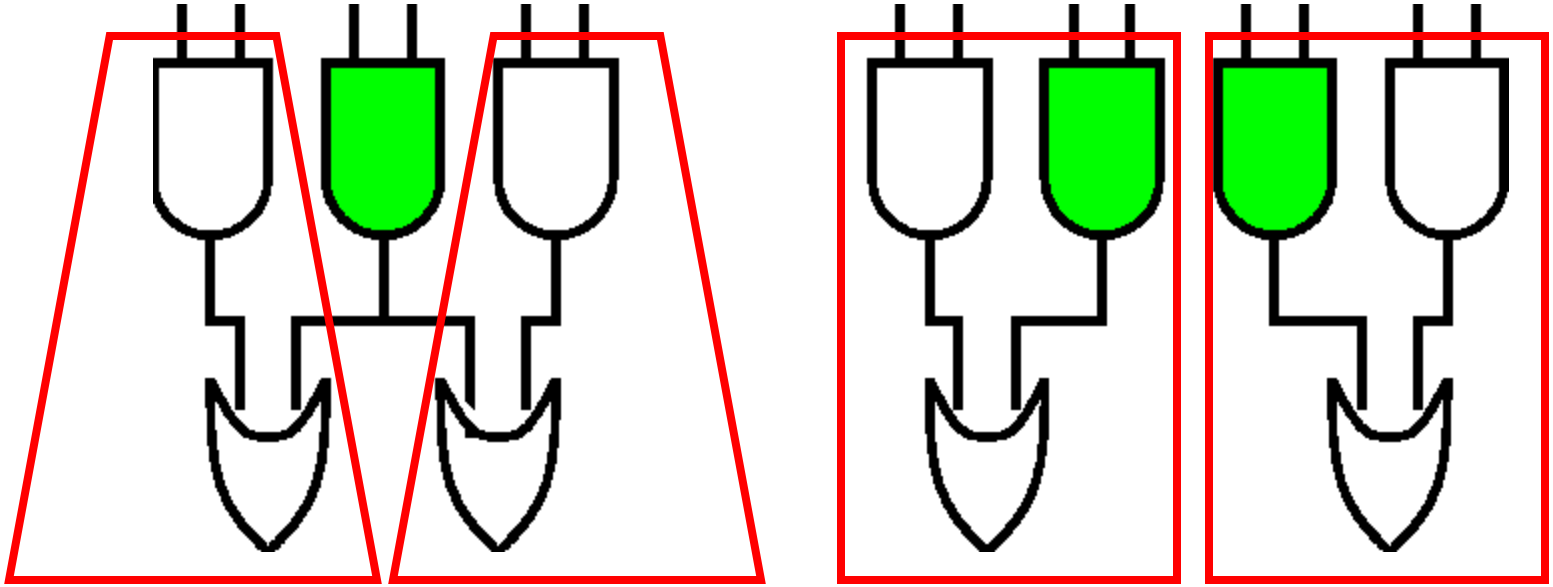




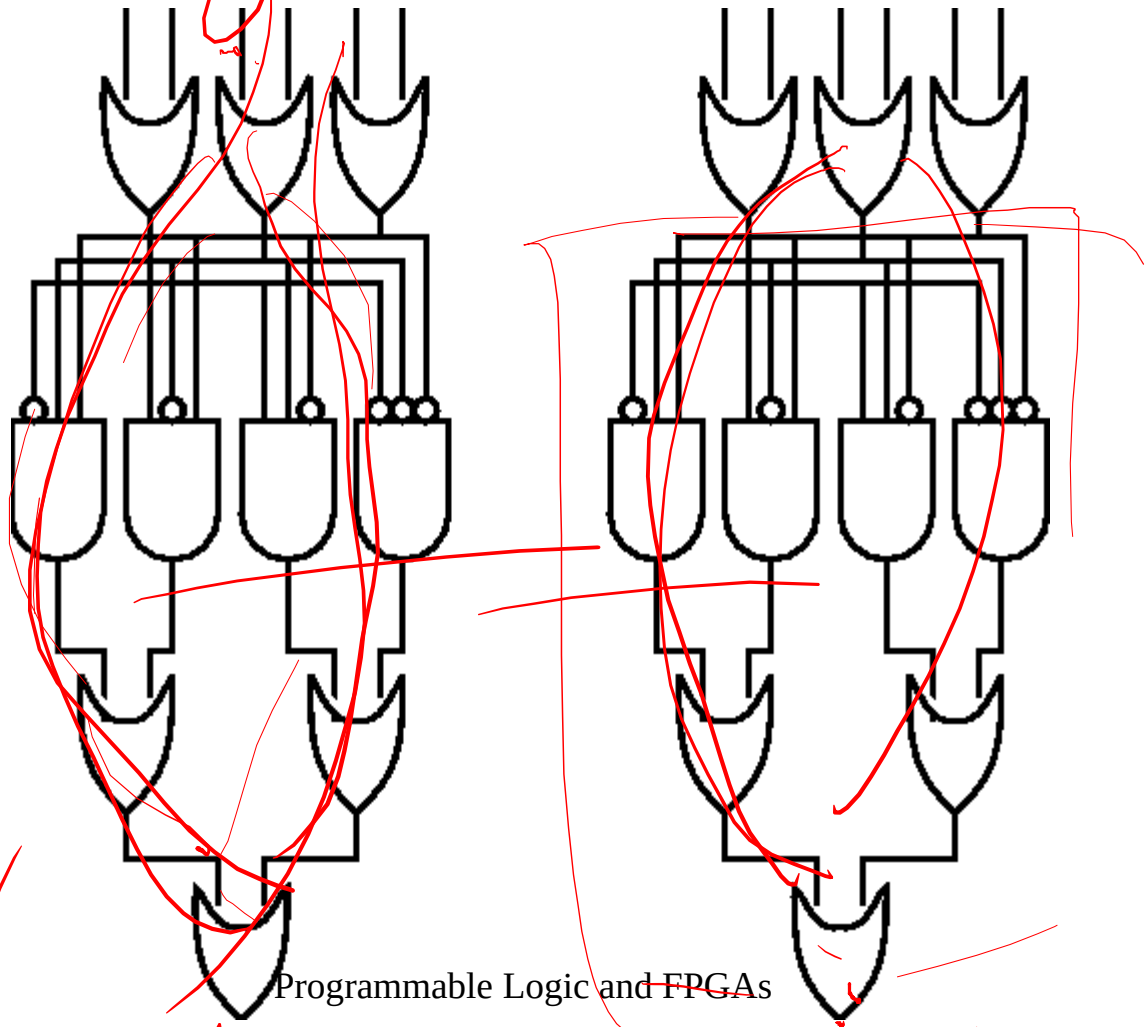
Fanout: Replication



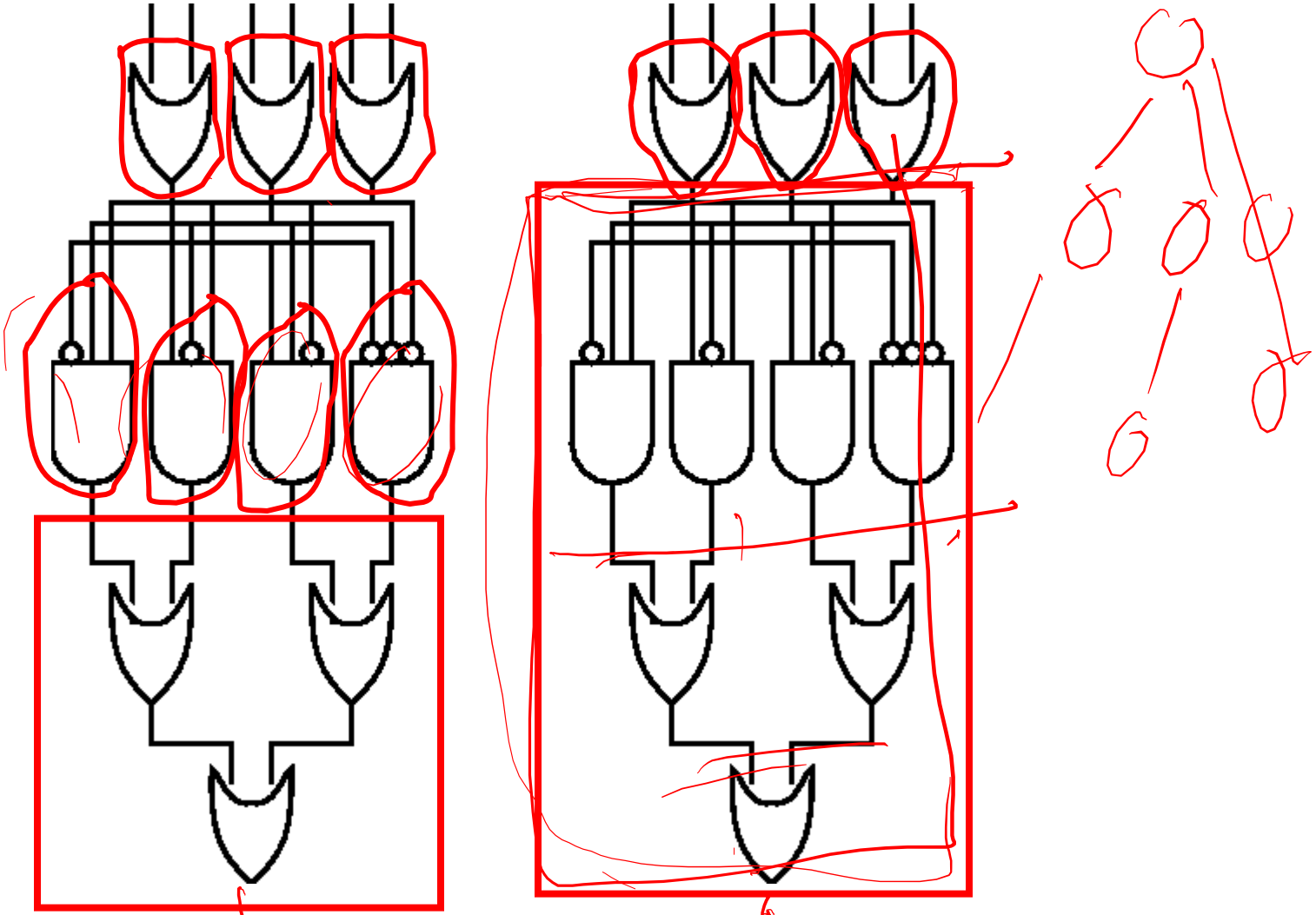
Fanout: Replication



Fanout: Reconvergence



Fanout: Reconvergence



CLB Mapping

Though direct mapping of technology independent circuit onto CLBs would involve function decomposition.

Alternatively, one can start from a circuit mapped onto LUTs and then pack them onto CLBs.

Thank You

N LUTs

m FFs

K LUTs

1 CLB { P FFs, T outputs }

16 bit reg

80

2 FFs

8 CLBs

T < K

16 LUT

T = K+1

T < K+P

max LUT, FF