

FPGA Design with VHDL

Justus-Liebig-Universität Gießen,
II. Physikalisches Institut

Ming Liu

Dr. Sören Lange

Prof. Dr. Wolfgang Kühn

ming.liu@physik.uni-giessen.de



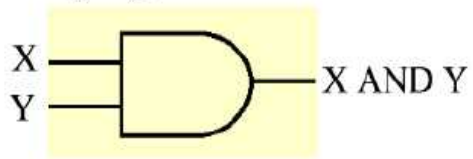
Lecture 1

- Digital design basics
 - Basic logic devices
 - Combinational circuits
 - Sequential circuits
- Programmable Logic Devices
 - IC classifications
 - FPGA architecture and technologies

Basic Logic Devices

AND

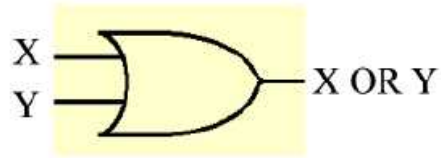
Logic gate



Truth table		
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

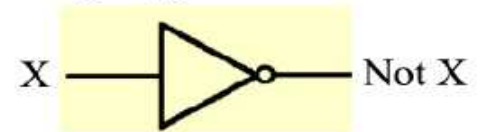
Logic gate



Truth table		
X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT

Logic gate

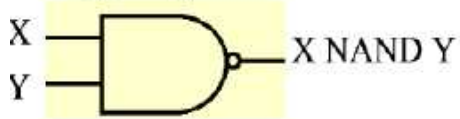


Truth table	
X	Not X
0	1
1	0

Basic Logic Devices

NAND

Logic gate

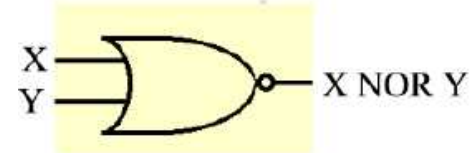


Truth table

X	Y	X AND Y	X NAND Y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NOR

Logic gate



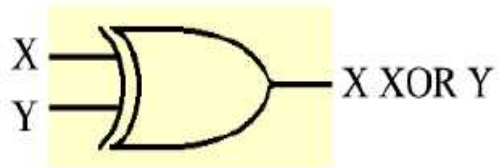
Truth table

X	Y	X OR Y	X NOR Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Basic Logic Devices

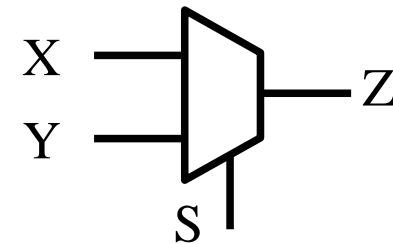
XOR

Logic gate



Truth table		
X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

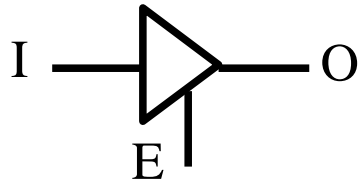
MUX



Truth table	
S	Z
0	X
1	Y

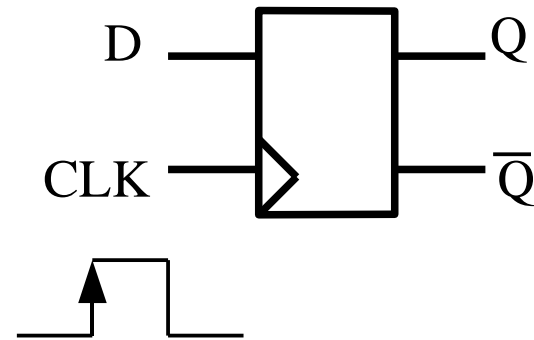
Basic Logic Devices


Tri-state



Truth table	
E	O
0	floating(Z)
1	I

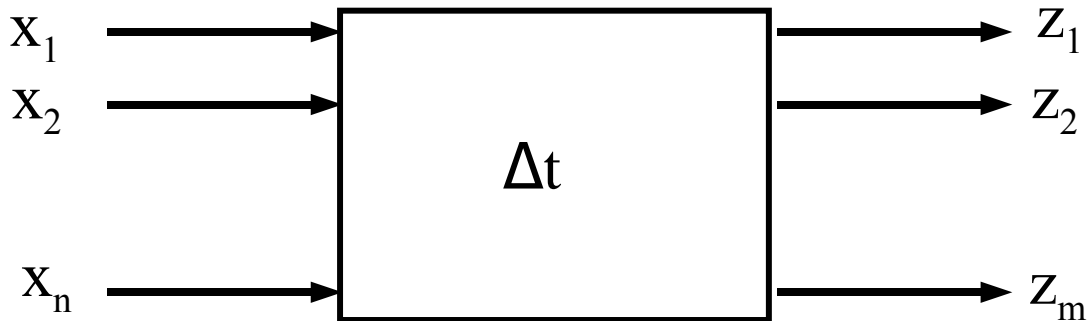
Flip-Flop



Truth table	
CLK	Q
	D
else	Q

Combinational Circuits

- Combinational circuits
 - Constructed with gate logics
 - Have no synchronous elements (FFs)
 - Have no concept of periodic timing
 - Outputs dependent only on inputs, after a delay time



$$Z_i = F_i(X_1, X_2, \dots, X_n)$$

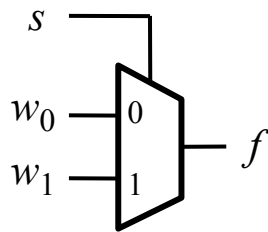


Combinational Circuits

- Examples:
 - Multiplexer
 - Adder
 - Multiplier
 - Divider
 - Decoder
 - Encoder
 - Asynchronous RAM
 -

Combinational Circuits: Example 1

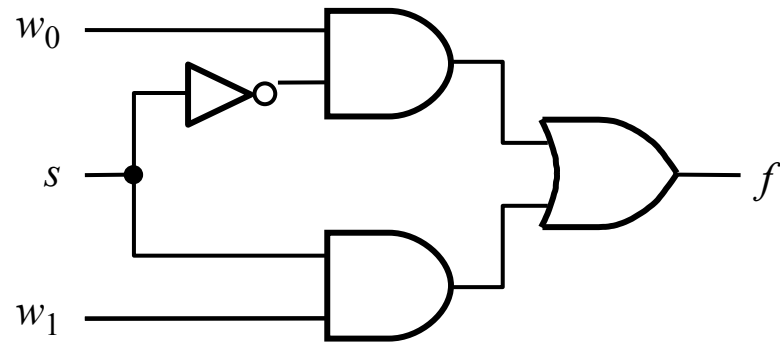
■ 2-to-1 Multiplexer



(a) Graphical symbol

s	f
0	w_0
1	w_1

(b) Truth table



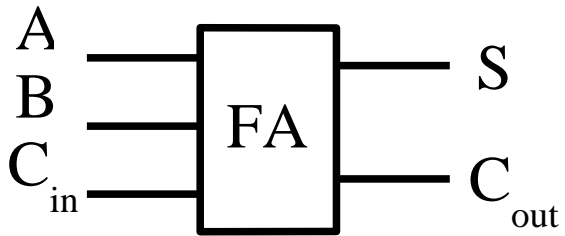
(c) Schematic

$$f = w_1 s + w_0 \overline{s}$$

(d) equation

Combinational Circuits: Example 2

- Full Adder (FA)



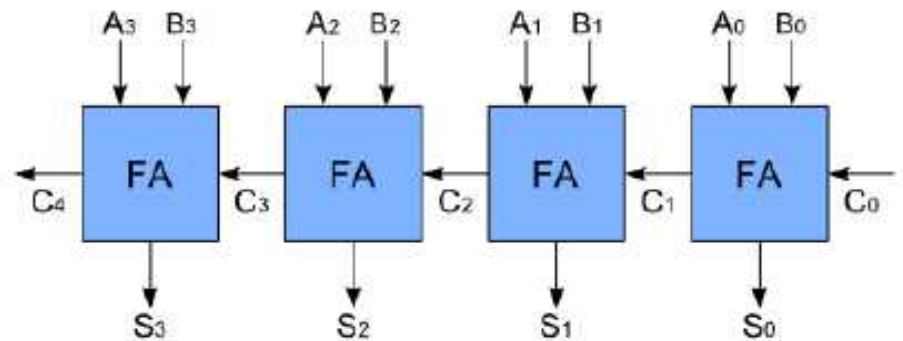
(a) Graphical symbol

Hidden for the lab!!!
Do it by yourselves!!!

Hidden for the lab!!!
Do it by yourselves!!!

(b) Truth table

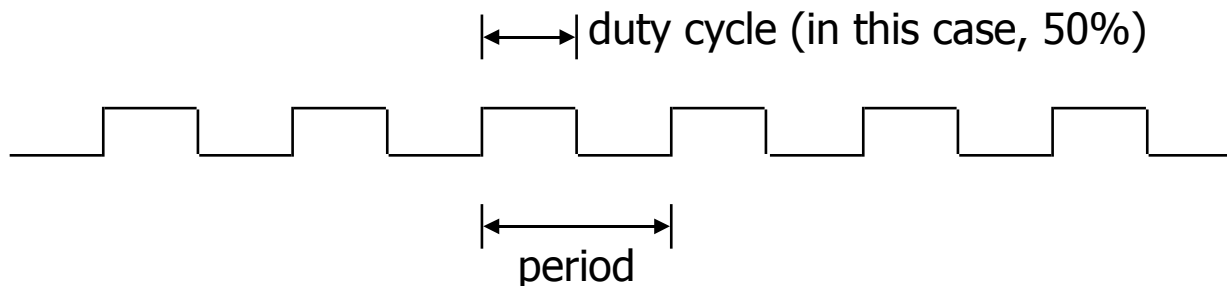
(c) Schematic



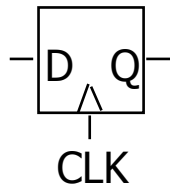
(d) Ripple Carry Adder

Sequential Circuits

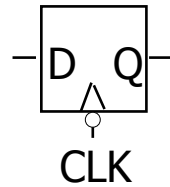
- Sequential circuits
 - Constructed with gate logics & synchronous elements (FFs)
 - Concept of periodic timing
 - Outputs updated at clock rising edge or falling edge
 - Important basics for pipelined processing
- Clocks are regular periodic signals
 - Period (T = time between ticks)
 - Frequency = $1/T$
 - Duty-cycle (time clock is high between ticks - expressed as % of period)



Synchronous Element (FFs/Registers)

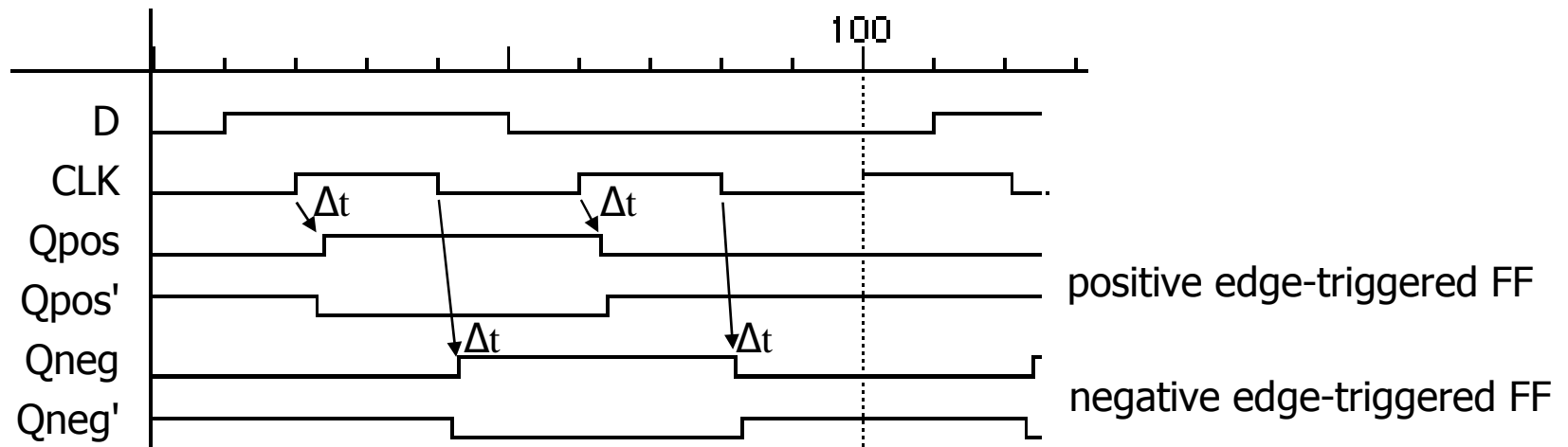


positive
edge-triggered
flip-flop

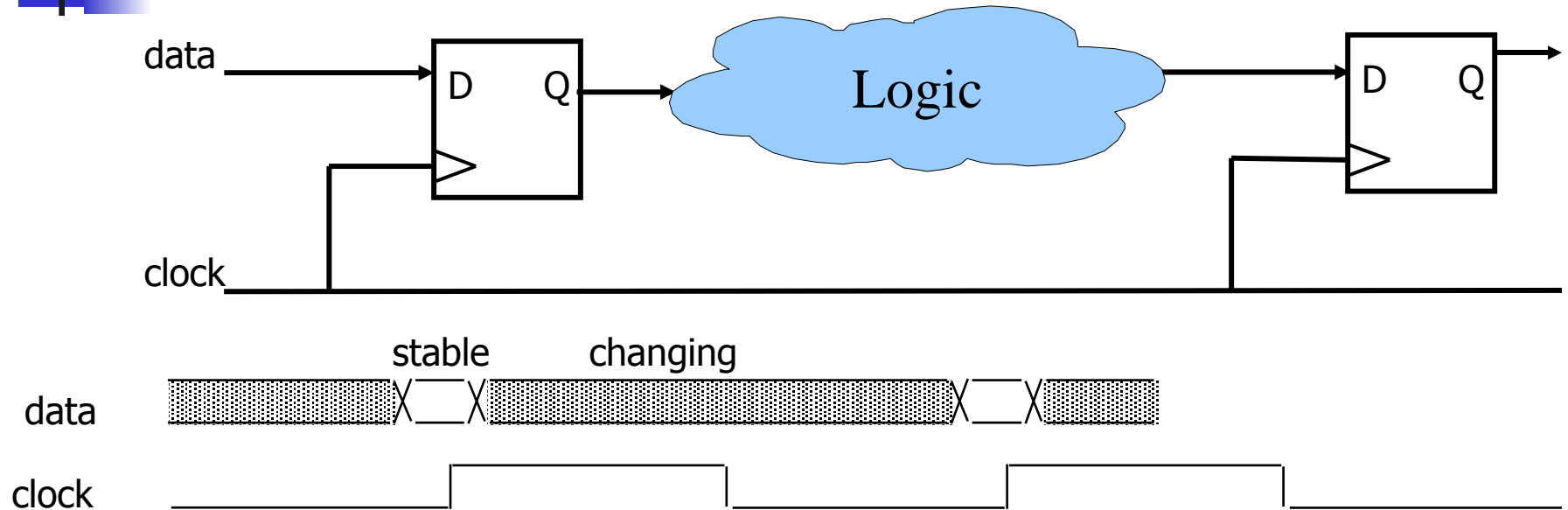


negative
edge-triggered
flip-flop

- Inputs sampled on clock rising/falling edge
- outputs change after a delay

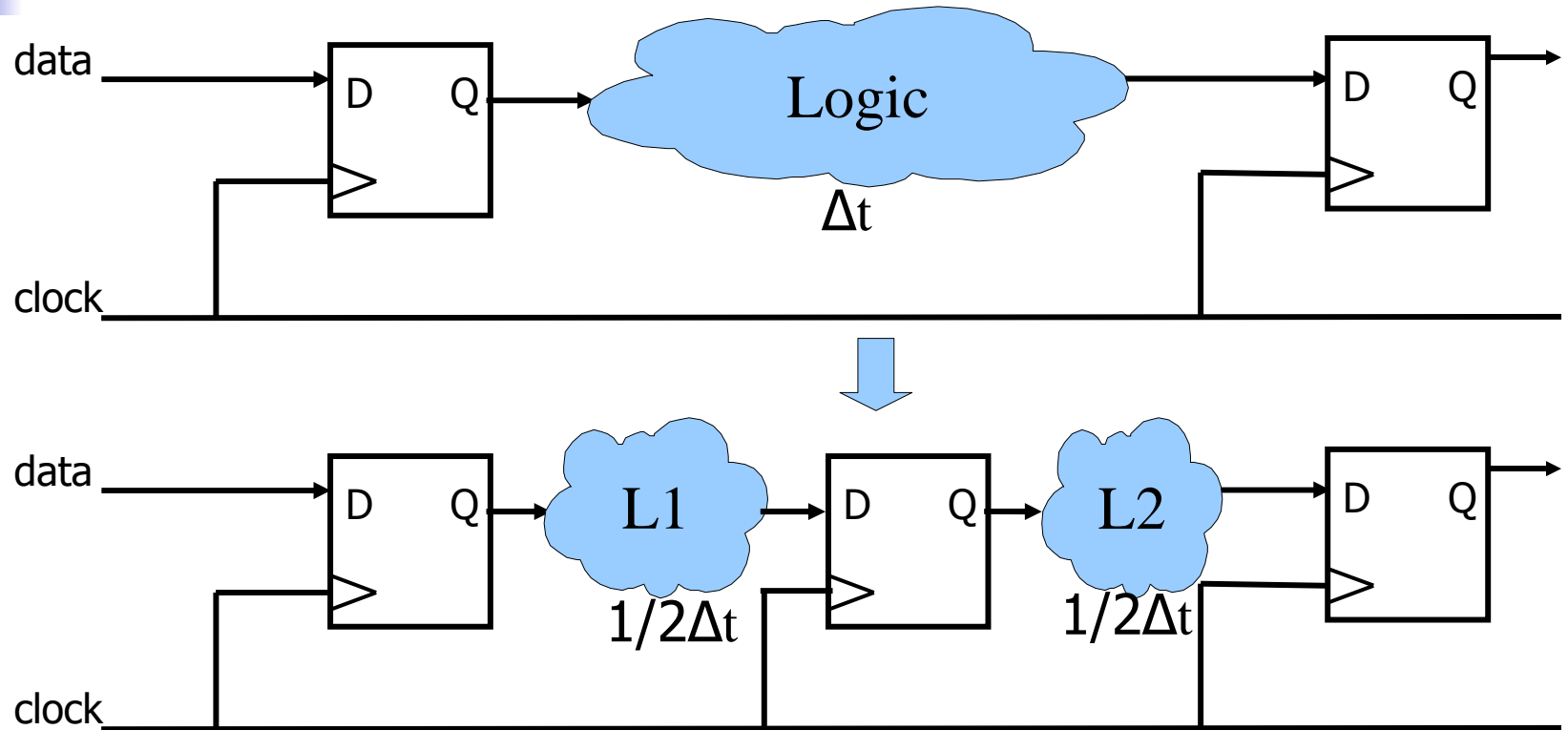


Timing Analysis



- Data is transferred from register to register
 - Combinatoric logic between the registers
 - **Critical Path** is the delay between two register levels
 - $f_{CLK} = 1/(T_{cp} + T_{ff})$
 - Register outputs are stable between clock cycles
 - No glitches on the register outputs

Pipeline



- For a pipelined design
 - According to $f_{CLK} = 1/(T_{cp} + T_{ff})$, f_{CLK} can be roughly doubled
 - One more clock cycle delay introduced
 - Computation throughput roughly doubled
 - One more register utilization



Synchronous vs. Asynchronous Designs

- Synchronous circuits (clocked)
 - Inputs are sampled and outputs changed in relation to a common reference signal (the clock)
- Asynchronous circuits (not clocked)
 - Inputs directly change outputs independently of a common reference signal (glitches a major concern)
 - Stay away from asynchronous designs !
(only if you can...)
- In this course, only synchronous circuits are concerned.



IC Classifications & Timeline

- In the early 80s :
 - Generic logic circuits (Example TTL: SN7400)
 - Complex applications assembled from basic building blocks: chips with few (< 10) hardwired logic functions
 - Many PCBs, interconnects, inflexibility, cost ...
 - Programmable PAL/GAL ...
- In the end 80s: FPGA invented by Xilinx, but only very limited capacity (<http://en.wikipedia.org/wiki/Fpga>)
- 90's: VLSI Circuits (ASICs) + “glue logics” (CPLD/FPGA)
- 00's: VLSI and PLD (especially FPGA)
- Nowadays, FPGA is large enough to host an entire system (System-on-an-FPGA), rather than only performing as glue logics.
- Programmable technologies are being merged with ASICs. FPGA-in-ASIC or ASIC-in-FPGA will be popular.

Comparison of different technologies



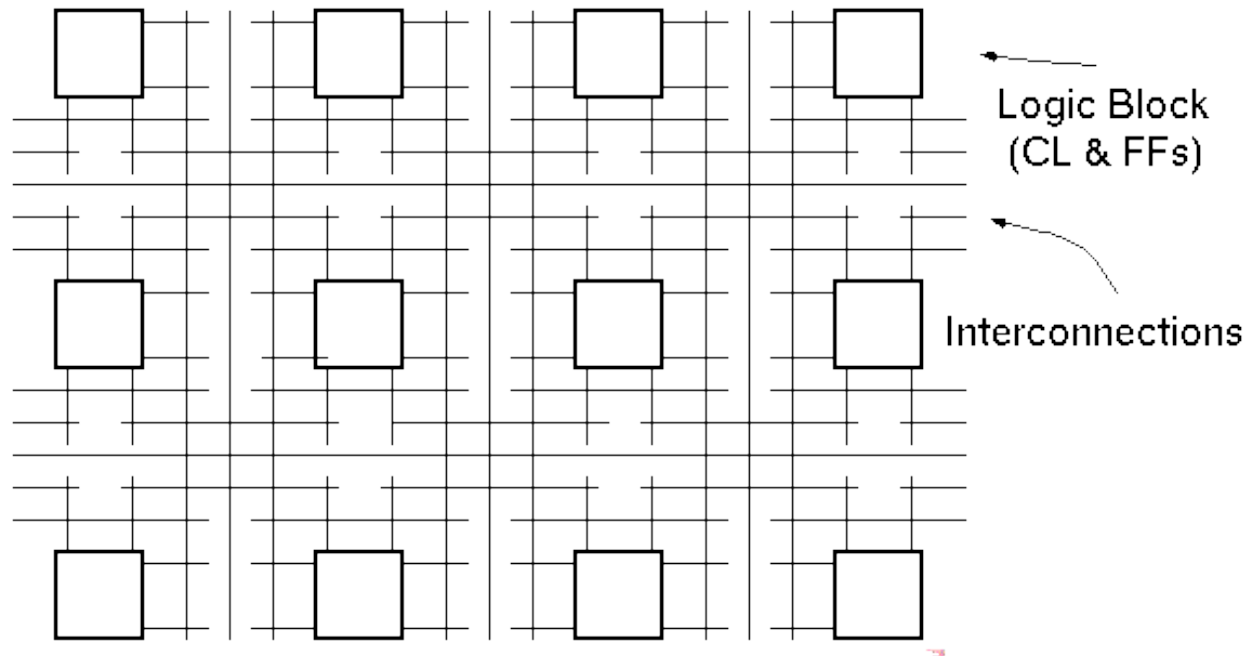
Technology	Performance/ Cost	Time until running	Time to high performance	Time to change code functionality
ASIC	Very High	Very Long	Very Long	Impossible
FPGA	Medium	Medium/ Long	Long	Long
DSP	High	Short/ Medium	Long	short/medium
Generic CPU/PC	Low-Medium	Short	Not Attainable	Very Short

The above conclusion is not really true. It depends on the real applications and cannot be easily called "good" or "bad"!!!

- Programmable Logic Device (PLD)
- A general term including all configurable devices
- CPLD (EPLD) + FPGA + PAL + GAL ...
- ROM-based, RAM-based, anti-fuse based
- RAM-based FPGA has large capacity and can be utilized in large-scale design. But it needs downloading configuration during power-on from non-volatile memories.
- ROM-based CPLD is small, but the configuration can be stored in non-volatile memories on-chip and needs not downloading during power-on.
- Anti-fuse based devices are mainly for aerospace and other radiation-aware applications.
- In this course, we discuss mainly normal FPGA designs

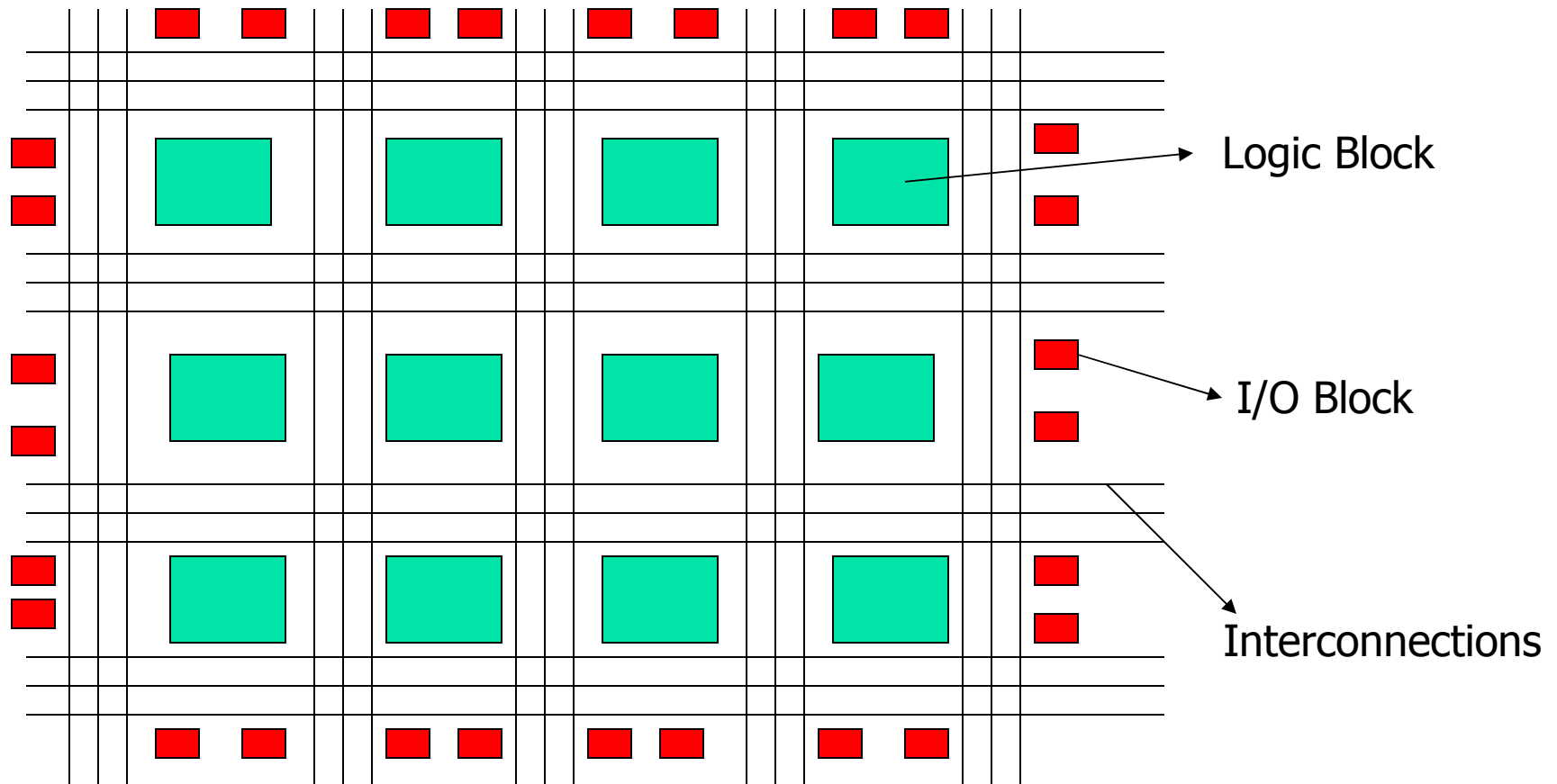
FPGA Overview

- Basic idea: 2D array of combination logic blocks (CL) and flip-flops (FF) with a means for the user to configure both:
 1. the interconnection between the logic blocks,
 2. the function of each block.

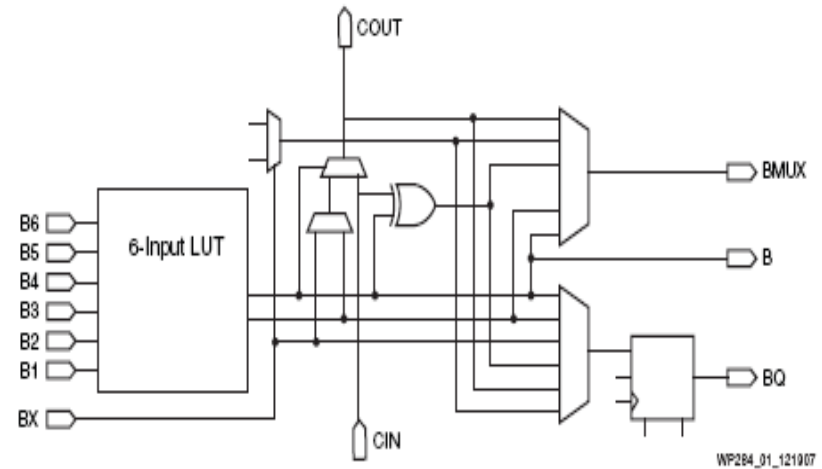
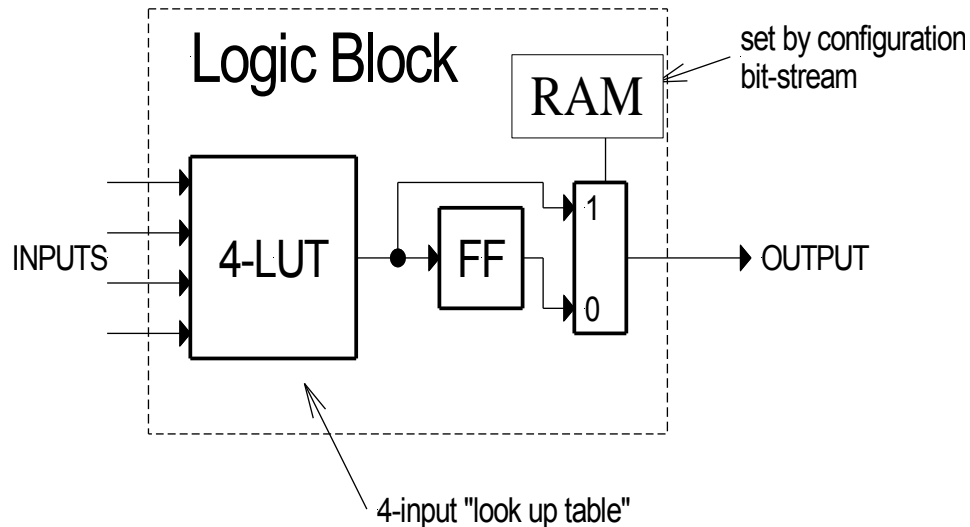


Simplified version of FPGA internal architecture

Structure of FPGA (Xilinx)



Simplified FPGA Logic Block



- 4-LUT vs. 6-LUT
 - implements combinational logic functions
- Register
 - optionally stores output of LUT
 - RAM determines output: register or LUT

LUTs as general logic gate

- An n-lut as a direct implementation of a function **truth-table**
- Each latch location holds value of function corresponding to one input combination

Example: 2-lut

INPUTS	AND	OR	
00	0	0	
01	0	1	
10	0	1	• • •
11	1	1	

Implements *any* function of 2 inputs.

How many functions of n inputs?

Example: 4-lut

INPUTS		
0000	$F(0,0,0,0)$	← store in 1st RAM bit
0001	$F(0,0,0,1)$	← store in 2nd RAM bit
0010	$F(0,0,1,0)$	←
0011	$F(0,0,1,1)$	←
0011		
0100	•	
0101	•	
0110	•	
0111		
1000		
1001		
1010		
1011		
1100		
1101		
1110		
1111		

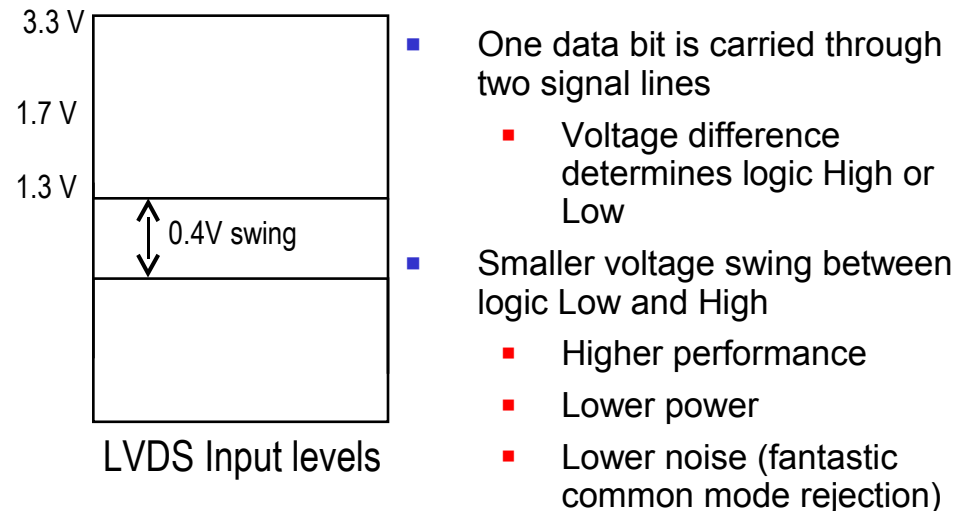
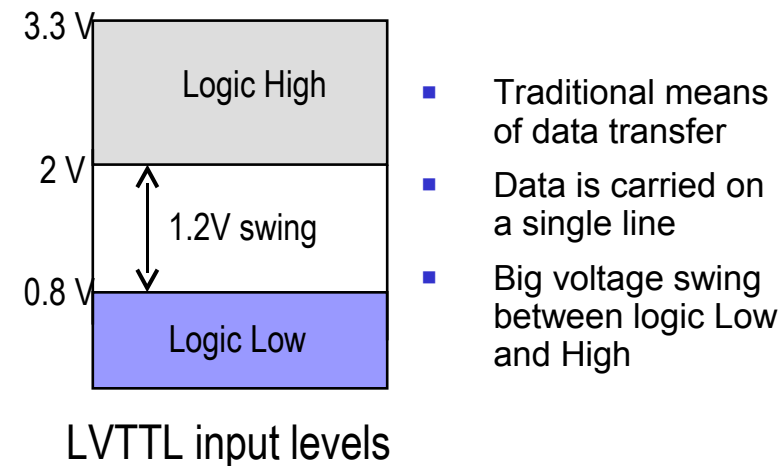
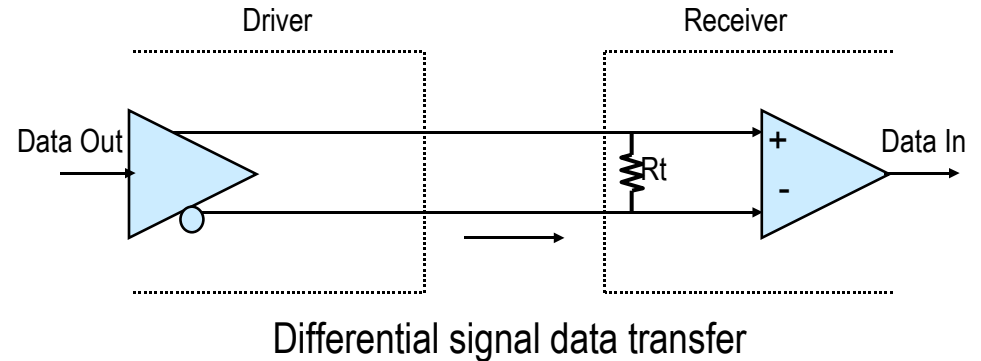
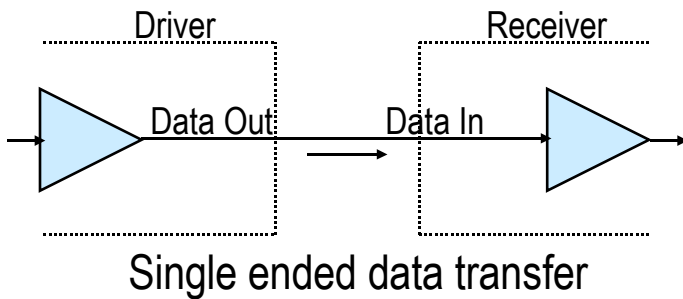


Advanced Programmable Resources

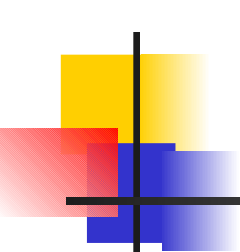
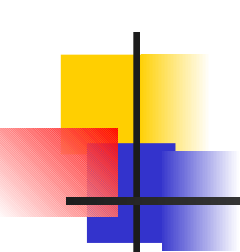
- Except for LUTs and FFs, other programmable resources include:
 - Block RAM (BRAM): dedicated RAM blocks on FPGAs. Can be used as small storage components for fast memory accessing.
 - DSP slices: multiplier and adder for DSP computation
 - Digital Clock Management (DCM): clock frequency synthesis
 - Hardcore IPs: processor, Ethernet MAC, RocketIO, ...
 - Refer to "Virtex-4 Libraries Guide for HDL Designs" for detailed primitives on Virtex-4 FPGA

Communication Channels

■ Single-end I/O (GPIO) vs. Differential I/O (LVDS)



-





Self-study

- Karnaugh Map (K-map)
 - Used to derive equations from truth tables
 - Can simplify equations for less gate utilization
 - http://en.wikipedia.org/wiki/Karnaugh_map
 - <http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karnaugh.html>
 - ...



References

- Wikipedia
- Virtex-4 User Guide
- Virtex-4 Configuration Guide