

Fall 2024: CSCI 181RT

Real-Time Systems in the Real World

Lecture 23

Thursday, November 14, 2024
Edmunds Hall 105
2:45 PM - 4:00 PM

Professor Jennifer DesCombes

Agenda

- Go Backs
- Discussion on Reading
- Lab #11 Review
- Programmable Array Logic (PAL)
- Complex Programmable Logic Device (CPLD)
- Field Programmable Gate Array (FPGA)
- Look Ahead
- Assignment
- Action Items

Go Backs

- General?
- Action Item Status
 - AI240910-2: Find recommended book on computer architecture.
 - AI240924-1: At what point as a development team grows does it make sense to have dedicated software and integration testers?
 - AI241024-1: Provide documentation on how to disable compiler optimization.
 - AI241107-1: Generate drawing showing location of Task Test Points on evaluation board.
 - AI241112-1: Send out Lecture Charts for Lecture 22 - OK to Close?

Discussion on Readings

- The Soul Of A New Machine
 - Chapters 7: La Machine

Lab 11 Review

- Use Newest Version of Template with RTOS ✓
 - Improved Partitioning with Functional & Data Privacy ✓
- Incorporate Hardware Input Capture Function
 - Use Input Capture #3
 - Programmable Matrix Matrix
 - Section 12 - Pin Mapping
 - Some Additional Code Was Distributed

Programmable Array Logic (PAL)

- Simple Array of Logic and Feedback Paths
- First Type of Programmable Logic
- Programmable via Dedicated Programmer
 - One Time Reprogrammable (Fuses)
- Can Be Very Fast
 - So Simple

Complex Programmable Logic Device (CPLD)

- Complex - More Than Most Standard Logic (of that time)
- Programmable via Dedicated Programmer (or JTAG)
 - Some Reprogrammable
 - Typically Not Programmable Once Installed
- Logic Design
 - Discrete Gates
 - Logical Grouping of Gates as Larger Blocks

Field Programmable Gate Array (FPGA)

- More Functionality (10^2 , 10^5 ??) Than CPLD
- Field Programmable
 - Can be Reconfigured in “the field”
 - Can be Easily Modified for Features and Options
- Gate Array
 - Discrete Gates
 - Logical Grouping of Gates as Larger Blocks

Field Programmable Gate Array (FPGA)

- Complex Topic?



University of Colorado

Boulder | Colorado Springs | Denver | Anschutz Medical Campus

Colorado Learning and Teaching with Technology



About COLTT

2024 Keynote Speaker

Program Committee

Conference Archive

Past Sponsors

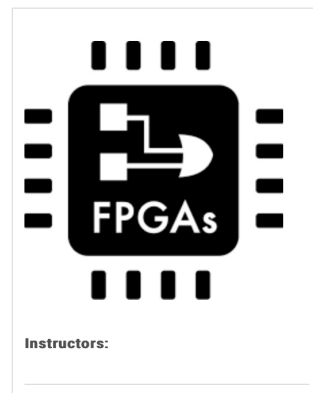
HOME | INTRODUCTION TO FPGA DESIGN FOR EMBEDDED SYSTEMS

Introduction to FPGA Design for Embedded Systems



About This Specialization

Programmable Logic has become more and more common as a core technology used to build electronic systems. By integrating soft-core or hardcore processors, these devices have become complete systems on a chip, steadily displacing general purpose processors and ASICs. In particular, high performance systems are now almost always implemented with FPGAs.



Field Programmable Gate Array (FPGA)

- Complex Topic?

ECEA 5360 Introduction to FPGA Design for Embedded Systems

1st course in the **FPGA Design for Embedded Systems Specialization**

Instructor: Timothy Scherr, MSEE, Senior Instructor

This course will give you the foundation for FPGA design in Embedded Systems. You will learn what an FPGA is and how this technology was developed, how to select the best FPGA architecture for a given application, how to use state of the art software tools for FPGA development and solve critical digital design problems using FPGAs. If you are thinking of a career in Electronics Design or looking at a career change, this is a great course to enhance your career opportunities.

Prior knowledge needed: Knowledge of assembly and C Programming, Digital Logic Design, and basic computer architecture. Students should have a first course in each of these subjects. The corresponding CU-Boulder courses are ECEN 2120/2350, ECEN 3100/3350, and ECEN 1030/1310/CSCI 1300. To be specific, you are expected to be able to perform tasks similar to designing sequential circuits using Karnaugh maps or Boolean equations.

Syllabus

+ [Week 1 | What's this programmable logic stuff anyway? History and Architecture](#)

+ [Week 2 | FPGA Design Tool Flow; An Example Design](#)

+ [Week 3 | FPGA Architectures: SRAM, FLASH, and Anti-fuse](#)

+ [Week 4 | Programmable logic design using schematic entry design tools](#)

+ [Week 5 | Final Exam](#)

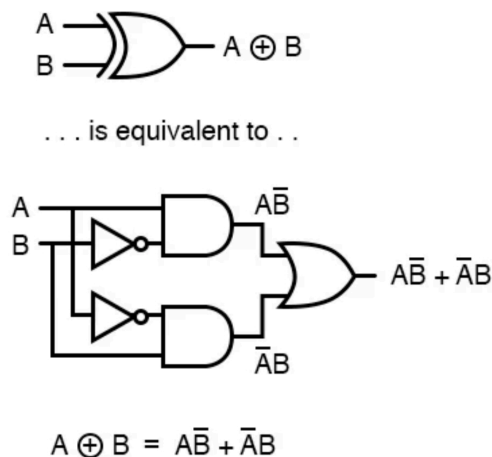
Field Programmable Gate Array (FPGA)

- Software Languages - VHDL and Verilog
 - VHDL (Very High Speed Integrated Circuit HDL)
 - Verilog
- Schematic Capture
 - No Longer A Reasonable Approach
- Other Software Tools
 - Compilers
 - Simulators
 - Debuggers







Logic - Gates

- Three Basic Functions
 - AND
 - OR
 - NOT
- Inverted Derivatives
 - NAND
 - NOR
- XOR -

Combined
Type

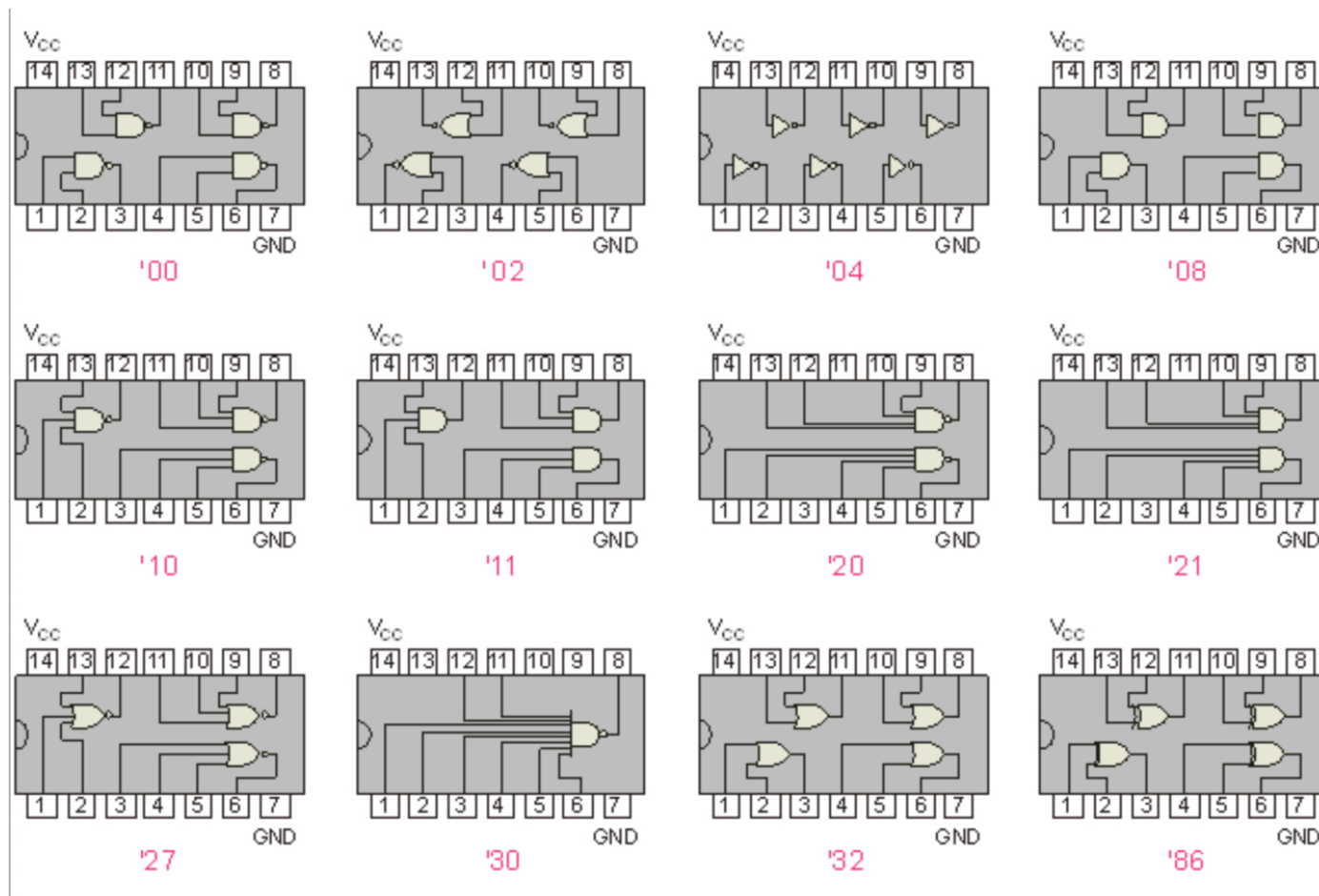


Digital Logic Gate Symbols

GATE	SYMBOL	NOTATION	TRUTH TABLE																		
<u>AND</u>		$A \cdot B$	<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A AND B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		OUTPUT	A	B	A AND B	0	0	0	0	1	0	1	0	0	1	1	1
INPUT		OUTPUT																			
A	B	A AND B																			
0	0	0																			
0	1	0																			
1	0	0																			
1	1	1																			
<u>OR</u>		$A + B$	<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A OR B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		OUTPUT	A	B	A OR B	0	0	0	0	1	1	1	0	1	1	1	1
INPUT		OUTPUT																			
A	B	A OR B																			
0	0	0																			
0	1	1																			
1	0	1																			
1	1	1																			
<u>NOT</u>		\overline{A}	<table><tr><th>INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>NOT A</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	INPUT	OUTPUT	A	NOT A	0	1	1	0										
INPUT	OUTPUT																				
A	NOT A																				
0	1																				
1	0																				
<u>NAND</u>		$\overline{A \cdot B}$	<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A NAND B</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		OUTPUT	A	B	A NAND B	0	0	1	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																			
A	B	A NAND B																			
0	0	1																			
0	1	1																			
1	0	1																			
1	1	0																			
<u>NOR</u>		$\overline{A + B}$	<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A NOR B</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		OUTPUT	A	B	A NOR B	0	0	1	0	1	0	1	0	0	1	1	0
INPUT		OUTPUT																			
A	B	A NOR B																			
0	0	1																			
0	1	0																			
1	0	0																			
1	1	0																			
<u>XOR</u>		$A \oplus B$	<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A XOR B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		OUTPUT	A	B	A XOR B	0	0	0	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																			
A	B	A XOR B																			
0	0	0																			
0	1	1																			
1	0	1																			
1	1	0																			

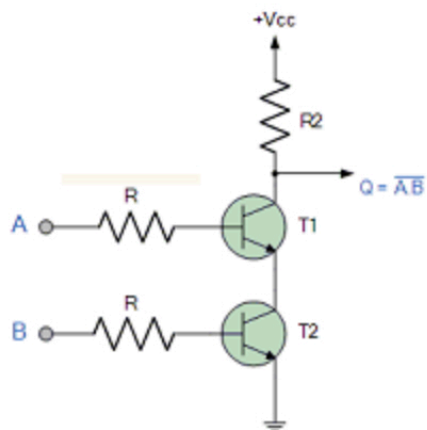
Logic - Gates

- LSI - Low Scale Integrated Circuits - 74 Series (also 54 - Mil)

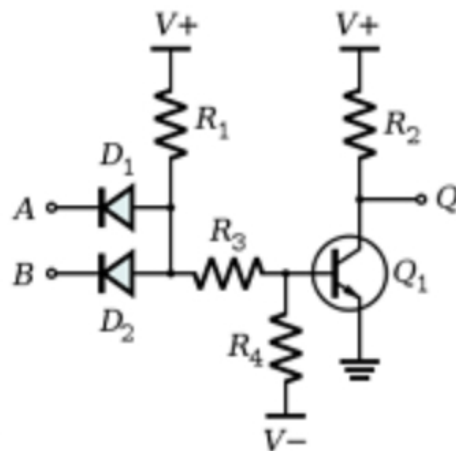


Logic - Gates

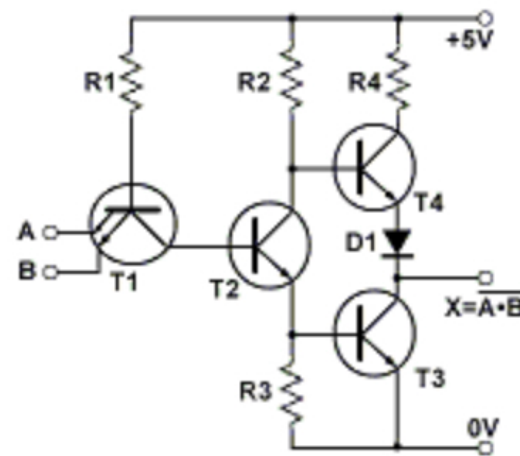
- Gates? How Do They Do That



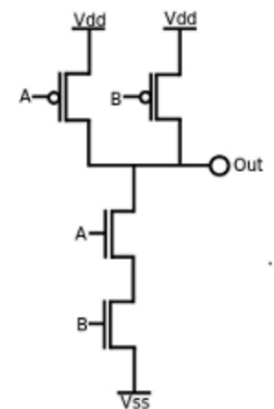
RTL NAND gate



DTL NAND gate



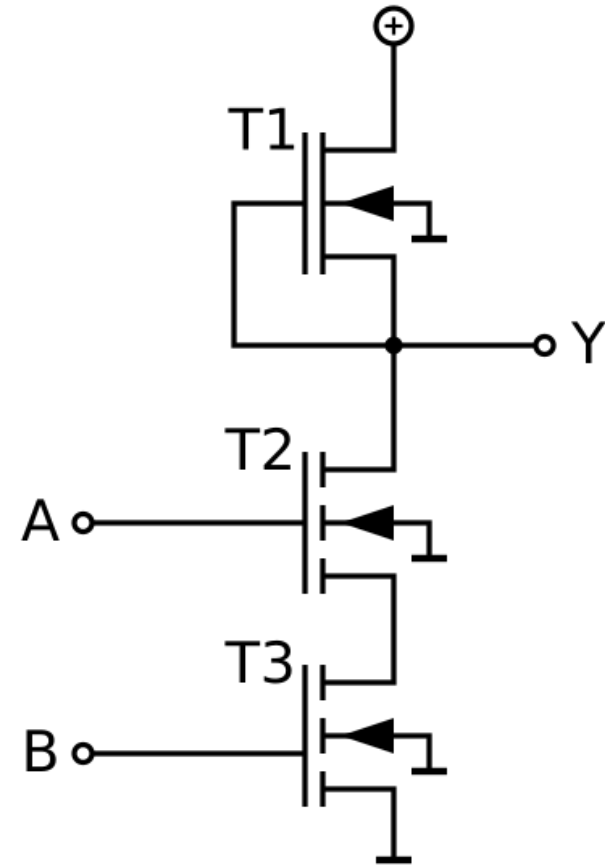
TTL NAND gate



CMOS NAND gate

Logic - Gates

- Gates? The Winner - NMOS NAND
- AND Gate is NAND + NOT
- NAND is Fastest Design

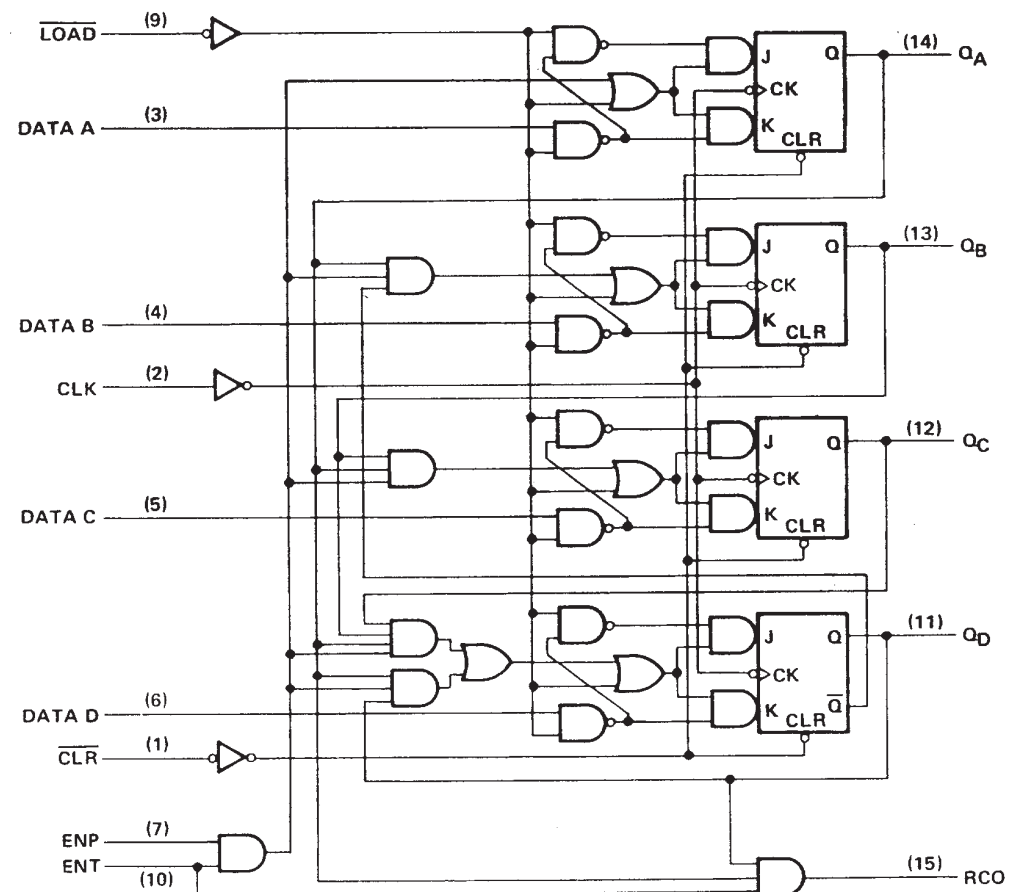


Logic - Gates

- 74160 - Decade Counter
- Integrated Into Single Package
- The Results (Qa-Qd) Are Brought Back as a Input
- Customizable?

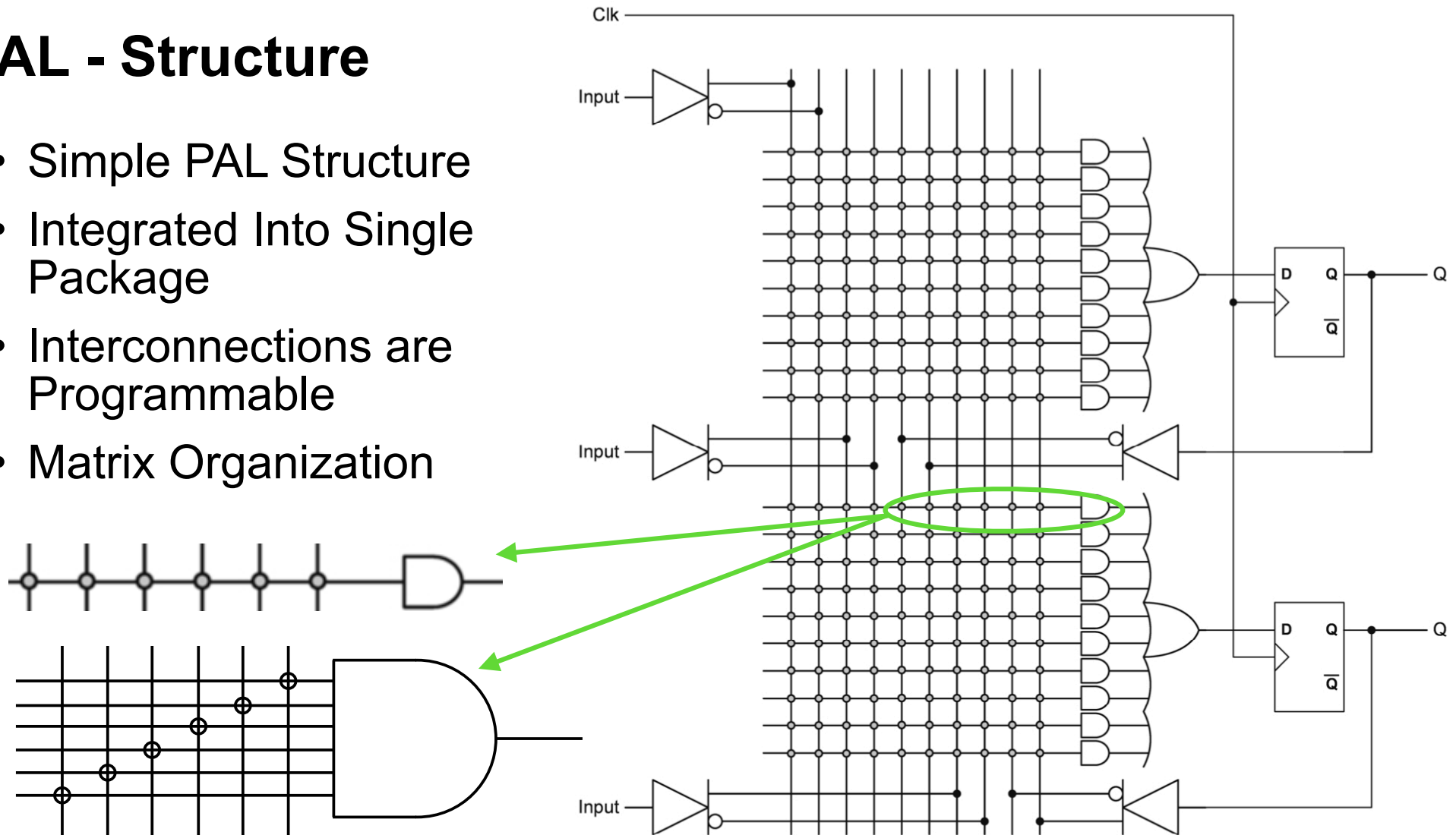
SN54160, SN74160 SYNCHRONOUS DECADE COUNTERS

SN54162, SN74162 synchronous decade counters are similar; however the clear is synchronous as shown for the SN54163, SN74163 binary counters at right.



PAL - Structure

- Simple PAL Structure
- Integrated Into Single Package
- Interconnections are Programmable
- Matrix Organization



CPLD - Structure

- Xilinx: XC9500 Series
- IO - Input, Output, Open Drain, Registers, Inverted
- Large Interterm Switch Matrix
- Functional Processing Blocks

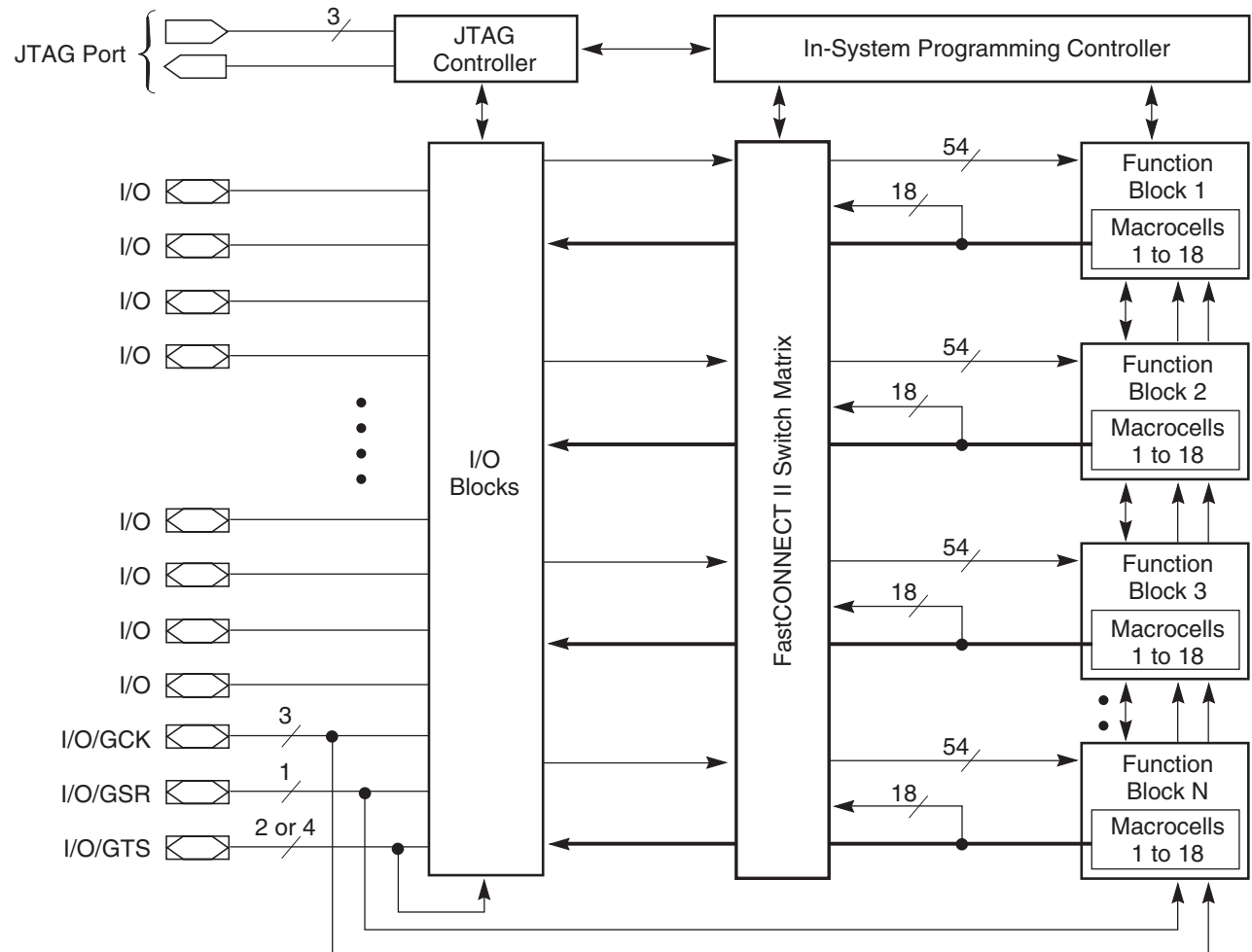
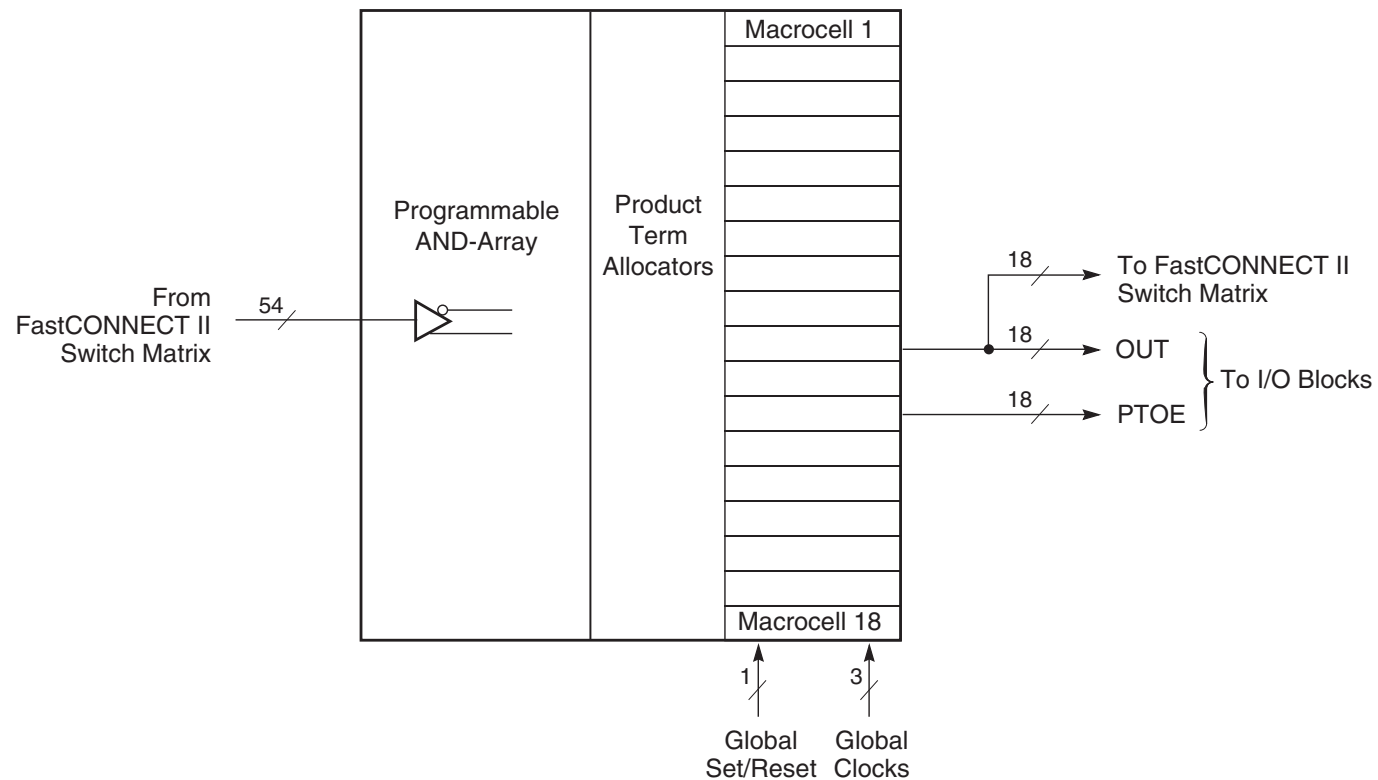


Figure 1: XC9500XL Architecture

Note: Function block outputs (indicated by the bold lines) drive the I/O blocks directly.

CPLD - Structure

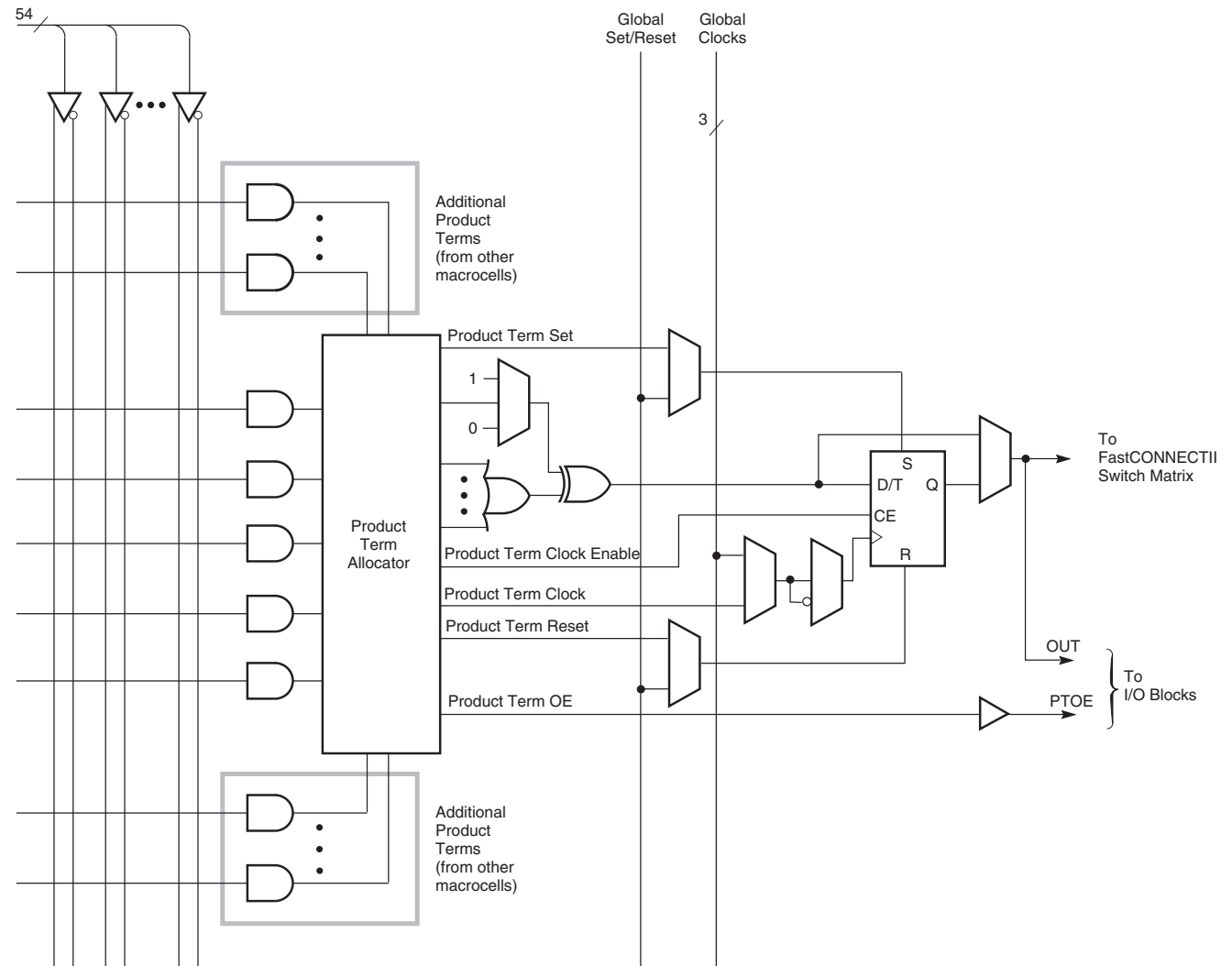
- Xilinx: XC9500 Series - Function Block Structure
- Product Term Allocators - Similar to PAL Matrix



DS054_02_042101

CPLD - Structure

- Xilinx: XC9500 Series - MacroCell
- Up to 288 in Large Package



DS054_03_042101

Figure 3: XC9500XL Macrocell Within Function Block

FPGA - Gates

- Xilinx: XC9500 Series - Capabilities by Package

Table 1: XC9500XL Device Family

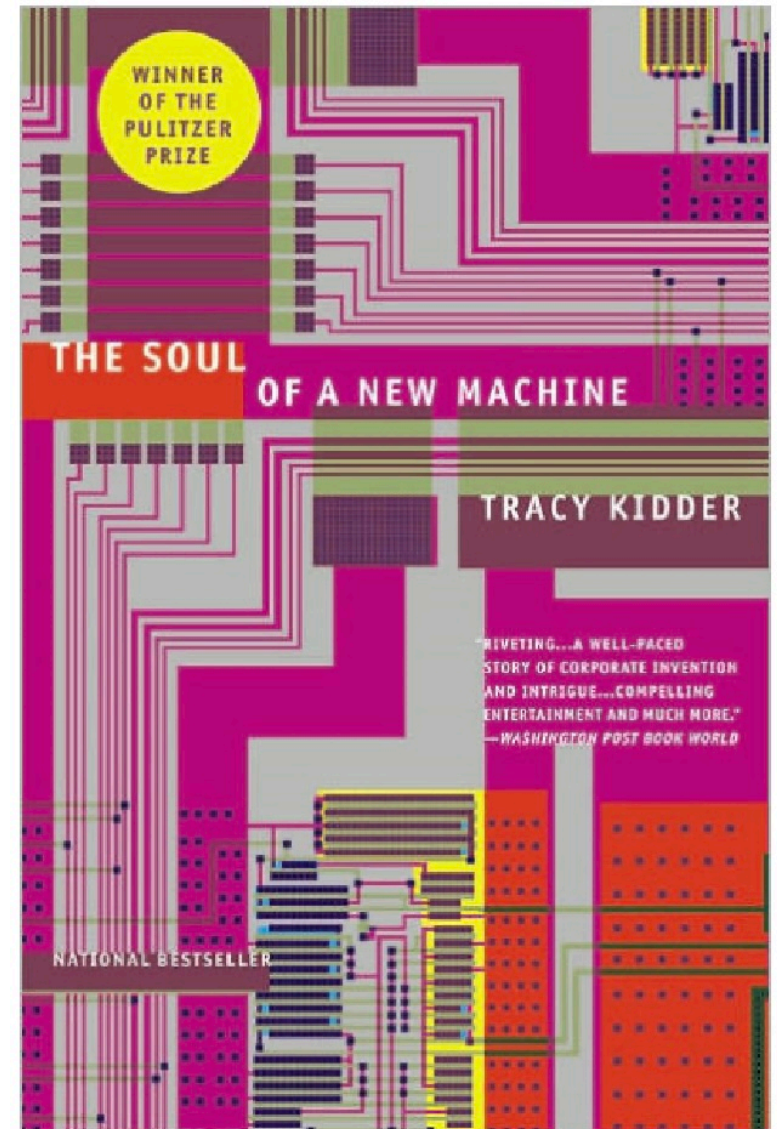
	XC9536XL	XC9572XL	XC95144XL	XC95288XL
Macrocells	36	72	144	288
Usable Gates	800	1,600	3,200	6,400
Registers	36	72	144	288
T _{PD} (ns)	5	5	5	6
T _{SU} (ns)	3.7	3.7	3.7	4.0
T _{CO} (ns)	3.5	3.5	3.5	3.8
f _{SYSTEM} (MHz)	178	178	178	208

Look Ahead

- Discussion on Reading
- More FPGA Functionality and Incorporation with RTOS
- Preview of Lab 12

Assignment - Readings

- The Soul Of A New Machine
 - Chapters 8 & 9: The Wonderful Micromachines, A Workshop
- Send Me Discussion Topics by 10:00 AM on Tuesday, November 19, 2024.



Assignment - Additional

- Grace-Bluegrass Basic Logic Gates.pdf
- Xilinx_CPLD_Architecture.pdf
- Many_types_of_FPGAs.pdf
- University of Wisconsin programmable-logic.pdf
- CERN Introduction to FPGA Design.pdf

Action Items and Discussion

AI#:	Owner	Slide #	Document	Action