INTRODUCTION TO FPGA PROGRAMMING

LESSON 03: BOOLEAN ALGEBRA, LOOK-UP TABLES AND IOBS

Dr. Davide Cieri¹

¹Max-Planck-Institut für Physik, Munich

September 2024



- Digital logic hides the pitfalls of the analogue world by mapping the infinite set of real values into two subsets corresponding to just 2 possible logic values: 1 and 0, or high and low voltages.
- With this assumption, we can use Boolean algebra to describe the operation of well behaved 0s and 1s in a circuit
- Instead of multiplication or division, Boolean algebra employs operations like AND, OR or NOT.

LOGIC GATES

- Logic Functions are represented in digital systems with *Logic gates*.
- Logic gates can be described with *truth tables*
 - A table listing all possible input and output combinations for a boolean algebra equation
- Logic Gates are implemented in VHDL using logic operators AND, OR, NOT, NAND, NOR and XOR.

 $A \leq X \text{ AND } Y;$

B <= X XOR Y;

(a)	Y	-[)'	(ANI X+)		(b))						(c) <u>×</u>	
		Х	Y	ХA	ND Y				Х	Y	XOR	r		x	NOT X
		0	0		0				0	0	0	_		0	1
		0	1		0				0	1	1			1	0
		1	0		0				1	0	1				
		1	1		1				1	1	1				
	(a)	-	Y	L		0—	NAND (X · Y)'		-	(b)	Y)	X	X NOR (X + Y	
				Х	Υ	X N/	ND Y					Х	Υ	X NOR Y	
				0	0		1					0	0	1	
				0	1		1					0	1	0	
				1	0		1					1	0	0	
				1	1		0					1	1	0	

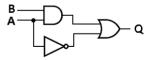


COMBINING GATES

- Gates can be combined to create different boolean equations
- VHDL follows the following logical operator order
 - AND > OR > NAND > NOR > XOR > XNOR
- Round parenthesis can be used to force a different order

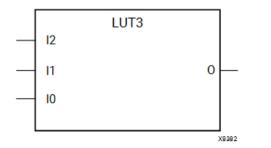
 $Q \iff A AND B XOR A$

Truth	Truth Table – A*B + A'										
Input A	Input B	Output Q									
0	0	1									
0	1	1									
1	0	0									
1	1	1									



THE LOOK-UP TABLE

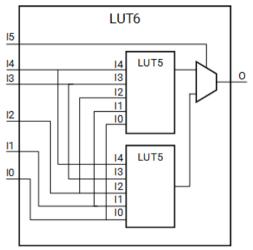
- On the FPGA, the are no physical logic gates that you can plug together to form a boolean algebra equation
- Look-Up Tables (LUTs) replace all functionalities of logic gates
- LUTs are devices that can be configured to implement any truth table
- They are defined by their number of inputs. E.g. LUT3



	Inputs	Outputs					
12	11		o				
0	0	0	INIT[0]				
0	0	1	INIT[1]				
0	1	0	INIT[2]				
0	1	1	INIT[3]				
1	0	0	INIT[4]				
1	0	1	INIT[5]				
1	1	0	INIT[6]				
1	1	1	INIT[7]				
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute							

XILINX 7-SERIES LUTS

- On 7-series Xilinx/AMD FPGAs, LUT6 are available
- More info, here



X10949

PIN CONSTRAINTS

- Logic signals in your design top module must be mapped to physical pins on the FPGA
- This ensures correct signal routing and interface with the external world
- If not specified, Vivado will place them randomly
 - It will fail, when creating a bitstream at Design Rule Check (DRC) because voltage standards are not defined
- Two options to constraint your design:
 - Using the I/O Planning graphic interface
 - With constraint files (.xdc for Vivado)

I/O PLANNING

- To access the I/O Planning, open the RTL analysis elaborated design
 - Then open Layout->I/O Planning
- The diagram shows the all the available pins on the device
- In the bottom panel, you can see the ports of your design, and can assign to them to a particular pin
 - Don't forget to define the IOSTANDARD
 - For our board, the standard is always LVCMOS33 (Low-Voltage CMOS 3.3V)



Tcl Console	le Messages Log Reports Design Runs Package Pins 1/O Ports x ?															
Q X + H H + H																
Name	Dir	rection		Board Part Pin	Board Part Inter	face I	Neg Diff Pair	Package Pin		Fixed	Bank	I/O Std		Vcco	Vref	I.
< 📾 All ports (50)																
🗸 🦻 BTN ((5) IN									~	14	LVCMOS33*		3.300		
🕑 BT	'N[4] IN							U18	~	~	14	LVCMOS33*	*	3.300		

XDC FILES

- What the GUI actually does is writing the constraint in .xdc file
- XDC files follow the Tcl semantic and contain list of commands to be executed by Vivado
- Tcl (Tool Command Language) is a powerful interpreted language with a simple syntax
 - Most IDE including Vivado have a Tcl console. All Vivado commands are written in Tcl.
 - Tcl is case-sensitive, on the contrary of VHDL
- Both timing and physical constraint can be written in an XDC¹
- AMD Xilinx user guide on Using Constraint here

¹You can use also .tcl file to specify your physical constraints. They follow the same syntax.

SPECIFYING PHYSICAL CONSTRAINTS

- Physical constraints are properties of any object in your design
- Properties are set using the set_property command
- Design interface are selected using the get_ports <port_name> command
 - Port name can also be a wildcard to set properties of multiple ports at the same time
- You can also define variables inside the xdc with the set <variable_name>
 <declaration> command

```
# set_property syntax
set_property <property> <value> <object_list>
# Example
set rst_port [get_ports {rst}]
# Variable content is accessed with the $ symbol like in bash
set_property PACKAGE_PIN A1 $rst_port
```

USEFUL XDC COMMANDS

- Some of these commands need to have the design open (RTL, Synthesis or Implementation)
- Each command has an help message. <command_name> -h

```
# get_ports: Get a list of ports in current design. If no argument, returns all ports
get_ports [<options>] [<port_name(s)>]
# list_property: List all the properties for a particular object
list_property [options] <object>
# list_property_value: Get a list of valid value for a specific property and object
list_property_value [options] <property> <object>
```

WHERE DO I GET INFORMATION ABOUT THE PIN MAPPING?

- How the FPGA pins are connected into your board is a choice of the board designer
- The FPGA developer should refer to the board schematic to check the pin assignment
- Basys3 board schematics: Page 6

	IC7A		
	BANK 14		
VGA_G3 D17		H19	VGA R1
LED6 U14		G19	VGA RO
	IO_25_14 IO_L4N_T0_D05_14	H17	VGA G1
	IO_L5P_T0_D06_14	G17	VGA G2
	IO_L5N_T0_D07_14	J19	VGA R2
	IO_L6N_T0_D08_VREF_14	517	VGA G0
	IO_L7P_T1_D09_14	118	VGA B3
	IO_L7N_T1_D10_14	1.18	VGA B1
	IO_L8P_T1_D11_14	K18	VGA B2
	IO_L8N_T1_D12_14	N18	VGA BO
	IO_L9P_T1_DQS_14	N19	VGA R3
	IO_L9N_T1_DQS_D13_14	P19	VGA HS
	IO_L10P_T1_D14_14	R19	VGA VS
	IO_L10N_T1_D15_14	M18	JC2
	IO_L11P_T1_SRCC_14	M19	JC8
	IO_L11N_T1_SRCC_14	L17	JC7
	IO_L12P_T1_MRCC_14	K17	JC1
	IO_L12N_T1_MRCC_14	N17	JC3
	IO_L13P_T2_MRCC_14	P17	JC9
	IO_L13N_T2_MRCC_14 IO_L14P_T2_SRCC_14	P18	JC4
		R18	JC10
	IO_L14N_T2_SRCC_14 IO_L16N_T2_A15_D31_14	W19	BTNL
	IO_L16N_12_A15_D31_14 IO_L17P_T2_A14_D30_14	T17	BTNR
	IO L17P 12 A14 D30 14 IO L17N T2 A13 D29 14	_T18	BTNU
	IO_L17N_12_A13_D29_14 IO_L18P_T2_A12_D28_14	U17	BTND
	IO_L18F_12_A12_D28_14 IO_L18N_T2_A11_D27_14	U18	BTNC
	IO_L19P_T3_A10_D26_14	V16	SW1
	IO L19N T3 A09 D25 VREF 14	V17	SW0
	IO L20P T3 A08 D24 14	W16	SW2
	IO L20P 13 A08 D24 14 IO L20N T3 A07 D23 14	W17	SW3
	IO L201 T3 DOS 14	V15	SW5
	IO L21N T3 DOS A06 D22 14	W15	SW4
	IO L22P T3 A05 D21_14	W13	SW7
	IO L221 T3 A04 D20 14	W14	SW6
	IO L22P T3 A03 D19 14	TU15	LED5
	IO L231 T3 A02 D18 14	U16	LEDO
	IO L24P T3 A01 D17 14	V13	LED8
	IO L24N T3 A00 D16 14	V13 V14	LED7
	10_22114_15_100_010_14		
		1	
	XC7A35T-1CPG236C		

SETTING PIN CONSTRAINTS FOR BASYS3

- We know have all the information to map the ports of our design to the physical pins on the Basys3 board
- For each port, we must define the PACKAGE_PIN and IOSTANDARD properties
 - Get the PACKAGE_PIN from the schematics
 - The IOSTANDARD for all Basys3 ports is LVCMOS33

Example
set_property PACKAGE_PIN U16 [get_ports {led}]
set_property IOSTANDARD LVCMOS33 [get_ports {led}]

LAB 04: WIRING SWITCHES TO LEDS

LAB 05: IMPLEMENT A FULL ADDER

LAB 06: HIERARCHICAL DESIGN (A RIPPLE CARRY ADDER IMPLEMENTATION)

The figures in these slides are taken from:

- Digital Design: Principles and Practices, Fourth Edition, John F. Wakerly, ISBN 0-13- 186389-4. ©2006, Pearson Education, Inc, Upper Saddle River, NJ. All rights reserved

- nandland.com

- docs.amd.com