- Today you will learn about hierarchical design of digital circuits
- The design will be done on 2 levels:
 - First you will design the submodules
 - Then the top module will be designed, and the submodules will be instantiated
- The PC is good example for hierarchical digital systems: the submodules are the processor, memory, VGA, etc. Each submodule has a well defined functionality, they are independent subsystems with unique behavior.
- In this case the top module is the motherboard, which connects the submodules, and integrates the subsystems (submodules) in to one highlevel system.

- The specification of the high level system of today's laboratory is the following: the circuit reads two hexadecimal numbers from the switches: one number from sw[3:0], and the other from sw[7:4].
- The two numbers (A and B), and their sum (A+B) will be shown on the 7-segment display of the system: one digit for A, one digit for B, and two digits for A+B (since the maximum value is F + F = 1E).
- To implement such system, we need the following components: multiplexer, decoder, hexadecimal to 7-segment converter.
- In addition, we need a circuit that continuously counts from 0 to 3. This is a sequential circuit, so it won't be studied in details right now.

• The design is the following:



- Let's start with the 7-segment display (in the top right corner). It has two group of inputs:
 - Inputs a b c ... g and dp: these inputs control the LEDs, dp is for the decimal point LED, it will not be used today.
 - Inputs DIGn: Digit selector inputs, the LED control inputs are forwarded to the selected digit.
- These are **negative logic inputs:** 0 input means true and 1 means false.
- **Example**: if the b and c inputs are 0, the others are 1 on the LED control input, and DIG2=0, the others are 1 on the digit selector input, than a 1 will appear on the second digit from the left.
- Note: we will design the circuit assuming positive logic, and we will invert the signals in the last step (as it is shown in the figure)

- The S1-S0 signals belong to the counter that counts from 0 to 3 repeatedly: 00, 01, 10, 11, 00, 01, 10, 11, 00,
- We use the decoder to decode this signal: the corresponding output of the decoder is set to 1 to select a digit.
- At the same time S1-S0 is the input of the multiplexor too: based on the value of S1S0, the multiplexor forward one of its 4 4-bit input.
- This value is converted to a 7-segment number, and the result of the conversion appears on the LED control inputs of the 7-segment display.
- Let's start the implementation.

• Launch the ISE design suite



• Create a new project



- Name: Digital_design_lab_4A
- Work on the D: drive
- Press Next

lew Project Wizard
lew Project Wizar

←Create New Project

Specify project location and type.

Enter a name, location	ons, and comment for the project		
Name:	Digital_design_lab_4A		
Location:	D:\Digital_design_lab_4A		
Working Directory:	D:\Digital_design_lab_4A		
Description:			
Select the type of to Top-level source typ	p-level source for the project		
HDL			~
More Info		Next >	Cancel

 \times

- Verify the settings:
 - Spartan3E
 - XC3S250E
 - TQ144
 - -4
- Press Next, than Finish

≽ New Project Wizard	ł
----------------------	---

←Project Settings

More Info

Specify device and project properties.

Property Name	Value	
Evaluation Development Board	None Specified	~
Product Category	All	~
Family	Spartan3E	~
Device	XC3S250E	~
Package	TQ144	~
Speed	-4	~
Top-Level Source Type	HDL	~
Synthesis Tool	XST (VHDL/Verilog)	`
Simulator	ISim (VHDL/Verilog)	`
Preferred Language	Verilog	`
Property Specification in Project File	Store all values	`
Manual Compile Order		
VHDL Source Analysis Standard	VHDL-93	`
Enable Message Filtering		

< Back

Next >

Cancel

 \times

- Right click on the label
- Select New source...



- Select Verilog module
- Name: mpx_4_4_1
- This will be a 4x4x1 multiplexor: It has 4 4-bit input
- Make sure "Add to project" is checked
- Press Next



←Select Source Type

Select source type, file name and its location.

 IP (CORE Generator & Architecture Wizard) Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor 	File name: mpx_4_4_1 Location: C:\Users\cloud\Digital_design_lab_4A
More Info	Add to project Next > Cancel

- Do **NOT** add the signals here, We will add them manually later
- Press Next, then Finish

>	New	Source	Wizard
---	-----	--------	--------

←Define Module

Specify ports for module.

Port Name	Directi	on	Bus	MSB	LSB	^
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				~

Х

- Add the following source code:
- Note: we use "output reg" instead of "output" in the module declaration
- This is needed to be able to use the always block
- As we have seen on the practice, this is a simpler and more overviewable way to describe the circuit's behavior.
- Use the "<=" operator for register assignments (instead of "=")
- Save all changes F

```
21 module mpx 4 4 1(
2.2
      input [1:0] s,
      input [15:0] num,
23
      output reg [3:0] out
24
25
      );
26
27
   always 0 (*)
28
      case (s)
          2'b00: out <= num[3:0];
29
30
          2'b01: out <= num[7:4];
31
          2'b10: out <= num[11:8];
32
          2'b11: out <= num[15:12];
33
          default: out <= num[3:0];</pre>
34
       endcase
35
36
   endmodule
37
```

• Add another new source file



- Verilog module
- Name: dec_2_4
- This will be the decoder
- Make sure "Add to project" is checked
- Press Next



- Leave the table blank
- Press Next, then Finish

🍃 New Source Wizard

←Define Module

Specify ports for module.

Module name dec_2_4						
Port Name	Direction		Bus	MSB	LSB	^
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				~

More Info

< Back Next >

Cancel

- Add the following source code:
- Mind the reg keyword after output in the header
- Save all changes 目

```
21 module dec 2 4(
22
      input [1:0] s,
23
      output reg [3:0] dig
24
      );
25
26 always @ (*)
27
      case (s)
28
         2'b00: dig <= 4'b0001;
         2'b01: dig <= 4'b0010;
29
         2'b10: dig <= 4'b0100;
30
         2'b11: dig <= 4'b1000;
31
32
         default: dig <= 4'b0001;</pre>
33
      endcase
34 endmodule
```

- Now we will test the decoder and the multiplexor
- First switch to Simulation mode

• Then right click on the label, and select New Source

	🍃 ISE	Project I	Vavigato	r (P.2 01 31	01 3) - C	:\Users [\]	(cloud)
	📄 File	e Edit	View	Project	Source	Pro	cess
	1	ا 🗟 🗲	9 &	1 X G		< 0	CI
	Design				•	→□₽	×
	T Vi	ew: 🔘 🗄	🔯 Imple	mentation		5imulatio	n 📷
	Ja Be	havioral					~
	E H	lierarchy					24
Project Na	vigator	(P.20131	013) - Cr	\Users\clo	oud\Digit	tal_desi	gn_lab
e Edit	View I	Project	Source	Proces	s Tool	s Wir	ndow
🖻 🖥 🕯		Χū			≝ »	Æ	P B
			+	→□₽×	4E	Q	11
'iew: 🔘 🔯	Implem	entation	🗩 🌆 S	imulation	₽ ∃	-	
ehavioral				\sim	_	9	
Hierarchy					24	10	11
🦳 😇 Digi	tal_desig	gn_lab_4/	4		_	11	11
🖹 🛄 xc3s	250e-4to	a <u>144</u>				10	11
	mnx	New S	ource			ΤZ	11
×	···P^ 6	Add So	ource			13	11

Desigi

- Select Verilog Test Fixture
- Name: mpx_4_4_1_tf
- Press Next

_			
	New	Source	Wizard

←Select Source Type

Select source type, file name and its location.

 BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor 	File name: mp×_4_4_1_tf Location: C:\Users\cloud\Digital_design_lab_4A
	Add to project
More Info	Next > Cancel

Х

- Select the mpx_4_4_1 circuit here
- Press Next, then Finish



←Associate Source

Select a source with which to associate the new source.



 \times

• Add the following code after the initial begin part

• Save all changes F

```
initial begin
    // Initialize Inputs
    s = 0;
    num = 0;
    // Wait 100 ns for global reset to finish
    #100;
    // Add stimulus here
    #100; s=2'b00; num=16'h048C;
    #100; s=2'b01;
    #100; s=2'b11;
end
```

• Left click on the mpx_4_4_1_tf module in the top left corner of the ISE

• Then right click on "Simulate Behavioral Model", and select "Rerun All"





- A window pops up with the simulation results. Press the "Zoom to Full View" button
- Right click on the out[3:0] signal, and select Radix->Hexadecimal
- Do the same for the s[1:0] and num[15:0] signals
- Check the simulation output





- Does the correct hexadecimal value appear on the output for each value of s[1:0]?
- Help: num = 0000010010001100 = 048C



- For example, for s=1 the output of the multiplexor is 8. Is that right?
- What is the output for s=0, s=2 and s=3?
- Close the simulator

• Now you will test the decoder. Right click on the label, and select New source...



- Name: dec_2_4_tf
- Don't modify the location
- Check "Add to project"
- Press Next

> New Source Wizard

←Select Source Type

Select source type, file name and its location.

BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Dackage VHDL Test Bench Embedded Processor	File name: dec_2_4_tf Location: C:\Users\cloud\Digital_design_lab_4A
More Info	Next > Cancel

- Select the dec_2_4 module
- Press Next, then Finish

🍃 New Source Wizard

←Associate Source

Select a source with which to associate the new source.



Х

Next >

Cancel

More Info

38

39

40

41

42

43

44

45

46

47

48

49

50

51

- We will test the module using a for loop
- Add the integer i; line before the initial begin part
- Don't forget to save your changes

```
integer i;
initial begin
   // Initialize Inputs
   s = 0;
```

// Wait 100 ns for global reset to finish
#100;

 Left click on the dec_2_4_tf module in the top left corner

• Then right click on "Simulate Behavioral Model", and select "Rerun All"





- Press the "Zoom to Full View" button
- Switch to hexadecimal number representation for the dig[3:0] and s[1:0] signals.



• Check the simulation results:



 The output is the following: s=00 -> dig=0001, s=01 -> dig=0010, s=10 -> dig=0100, s=11 -> dig=1000. Is that correct?

• Close the simulator

• Switch back to implementation mode

• Right click on the label, and select New Source...





- Name: hex7_seg
- Don't modify the location
- Check "Add to project"
- Press Next

≽ New Source Wizard

←Select Source Type

Select source type, file name and its location.

╡ ╏ ┚ て い て い 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、	BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor	File name: hex_7seg Location: C:\Users\cloud\Digital_design_lab_4A
Mo	ore Info	Next > Cancel

 \times

- Leave the table blank
- Press Next, then Finish

≽ New Source Wizard

←Define Module

Specify ports for module.

Module name hex_7seg						
Port Name	Directior	n	Bus	MSB	LSB	^
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				
	input	\sim				~

More Info

Next >

< Back

Cancel

- Add the following code
- Mind the reg keyword

- Copy and paste the code from the next slide above endmodule
- This will be the implementation of the hexadecimal to 7-segment converter, the output is defined for every possible input in the body of a case statement
- Think about it: otherwise we would have to design a 4-input combinational circuit for every LED, minimize the Boolean functions, and implement each

always @ (*)

case (hex)

4'b0000 : seg <= 7'b0111111; // 0

4'b0001 : seg <= 7'b0000110; // 1

4'b0010 : seg <= 7'b1011011; // 2

4'b0011 : seg <= 7'b1001111; // 3

4'b0100 : seg <= 7'b1100110; // 4

4'b0101 : seg <= 7'b1101101; // 5

4'b0110 : seg <= 7'b1111101; // 6

4'b0111 : seg <= 7'b0000111; // 7

4'b1000 : seg <= 7'b1111111; // 8

4'b1001 : seg <= 7'b1101111; // 9

4'b1010 : seg <= 7'b1110111; // A

4'b1011 : seg <= 7'b1111100; // b

4'b1100 : seg <= 7'b0111001; // C

4'b1101 : seg <= 7'b1011110; // d

4'b1110 : seg <= 7'b1111001; // E

4'b1111 : seg <= 7'b1110001; // F

default : seg <= 7'b0000000; // 0

endcase

- Save all changes 🗐
- Add another source to the project



This is the "div" clock divider, a sequential circuit that counts from 0 to 3

- Right now we do not go into the details, since you have not heard about sequential circuits
- Name: div
- Don't modify location
- Check "Add to project"
- Press Next

≽ New Source Wizard

←Select Source Type

Select source type, file name and its location.

 Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor 	BMM File ChipScope Definition a Implementation Const IP (CORE Generator & a MEM File	and Connection File raints File Architecture Wizard)	
Verilog Module div Verilog Test Fixture Location: VHDL Module C:\Users\cloud\Digital_design_lab_4A VHDL Package VHDL Test Bench Embedded Processor Image: Comparison of the second	 Schematic User Document 		File name:
Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor	Verilog Module		div
VHDL Library C:\Users\cloud\Digital_design_lab_4A VHDL Package VHDL Test Bench Embedded Processor	W Verilog Test Fixture		Location:
VHDL Package VHDL Test Bench Embedded Processor			C//Userc/sloud/Digital_decign_lab_40
VHDL Test Bench Embedded Processor	VHDL Package		
Embedded Processor	🔚 VHDL Test Bench		
	🜌 Embedded Processor		
	_		
Add to project			Add to project

More Info

Next > Cancel

×

- Leave the table blank
- Press Next, then Finish

≽ New Source Wizard

←Define Module

Specify ports for module.

Module name div					
Port Name	Direction	Bus	MSB	LSB	^
	input 🔹	-			
	input	-			
	input •	-			
	input ·	I			
	input 🔹	-			
	input 🔹	-			
	input 🕚	-			
	input 🕚	-			
	input	I			
	input	I			
	input	~ 🗆			~

More Info

Next >

< Back

Cancel

 \times

• Add the following source code

```
21 module div(
22
      input clk,
      output reg [1:0] s
23
24
     );
25
26
     always @ (posedge clk)
27
     begin
28
         s <= s + 1;
29
      end
   endmodule
30
```



 Now you can implement the top module, that connects all submodules together



- Name: Top
- Don't modify location
- Check "Add to project"
- Press Next

🍃 New Source Wizard

←Select Source Type

Select source type, file name and its location.

 BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor 	File name: Top Location: C:\Users\cloud\Digital_design_lab_4A Image: C:\Users\cloud\Digital_design_lab_4A
More Info	Next > Cancel

 \times

- Table: blank
- Next + Finish

>	New	Source	Wizard
---	-----	--------	--------

 \times

←Define Module

More Info

Specify ports for module.

Port Name	Direction	Bus	MSB	LSB	1
	input 🗸				
	input 🗸				
	input 🗸				
	input 🗸	· 🗆			
	input 🗸	· 🗆			
	input 🗸 🗸	· 🗆			
	input 🗸 🗸	· 🗆			
	input 🗸 🗸	· 🗆			
	input 🗸 🗸	· 🗆			
	input 🗸 🗸	· 🗆			
	input 🗸	· 🗆			~

- First add the declaration of the module:
- The "_n" tag after seg and dig means that these are negative logic signals:

```
21 module Top(
22 input [7:0] sw,
23 output [7:0] seg_n,
24 output [3:0] dig_n,
25 input clk
26 );
27
28
29 endmodule
```

Then add the following wires:

 a, b (for sw[3:0] and sw[7:4])
 s (for the output of the div module)
 dig (digit selector for the display)
 seg (LED control input for the display)
 num (multiplexor input)
 out (multiplexor output)

```
21 module Top(
      input [7:0] sw,
22
      output [7:0] seg n,
23
      output [3:0] dig n,
24
25
      input clk
26
      );
27
28
      wire [3:0] a;
29
      wire [3:0] b;
30
      wire [1:0] s;
31
      wire [3:0] dig;
      wire [3:0] out;
32
33
      wire [15:0] num;
34
      wire [7:0] seq;
35
36
   endmodule
37
\sim \sim
```

- Add the following assigments:
- Note that a+b will be connected to the lower 8 inputs of the multiplexor
- Since dig_n and seg_n are negative logic signals, we invert the values of dig and seg before the assignment.

```
34
      wire [7:0] seq;
35
36
      assign a = sw[7:4];
      assign b = sw[3:0];
37
38
      assign num[15:12] = a;
      assign num[11:8] = b;
39
      assign num[7:0] = a + b;
40
      assign dig n = ~dig;
41
      assign seg n = ~seg;
42
43
44 endmodule
```

- Note: dig and seg are 4 and 8 bits long.
 You can apply the ~ operator on N-bit wide wires, it will invert all bits
- For example, if x=1001, then $\sim x=0110$.

- The last step is the instantiation of the submodules: we need to define which wires of the Top module do we connect to the inputs and outputs of a given submodule.
- We also have to give a name for the instantiated submodule, since we can have multiple modules of the same type.
- Imagine you have a mod module with in input and out output. You want to connect the x wire to the input of the module, and y to the output. The syntax of the instantiation is: mod mod1 (.input(x), .output(y));
- Now mod1 is the name of the current instance

• Instantiate your submodules in the following way:

• Save all changes F

• After the instantiation, you will see that all submodules are organized under the Top module:



• The ISE indents the submodules under Top to show that they are on a lower hierarchical level

- Now we will implement and download the module to the FPGA board. A file has been prepared for this purpose, you can download it using the following link: <u>download</u>
- If the browser opens the file instead of displaying the download dialog window: right click, and select "Save as..."
- Download the file, and save it to the current working directory of your project (D:\Digital_design_lab_4A).
- If you are not sure, you can check it in the title bar of the ISE (top of the ISE window)

> ISE Project Navigator (P.20131013)	- C:\Users\cloud\Digital_design_lab_3\Digital_design_lab_3.xise - [BCD.v]	-		×
📄 File Edit View Project Sour	ce Process Tools Window Layout Help		-	Б×
🗋 ờ 🖥 🕼 🕼 🖌 🗅 🗅	×სიი∣» 🥬 ၉၉ இ 🖉 💽 📑 🗄 🖬 🖙 🥬 🥙 🔽 📌 💡			
P				_

 Go back to the ISE, right click on the "xc3s250e-4tq144" label and select "Add copy of source"



• Browse the file you have downloaded, select it and press Open:

^	Name	Date modified	Туре	Size	
		2019, 10, 01, 16:26	File folder		
	📊 ipcore_dir	2019, 10, 01, 13:41	File folder		
	📊 iseconfig	2019, 10, 01, 13:40	File folder		
	📊 isim	2019, 10, 01, 16:06	File folder		
	📊 xst	2019, 10, 01, 16:25	File folder		
	📄 BCD.ngc	2019, 10, 01, 16:27	NGC File	4 KB	
	BCD.v	2019, 10, 01, 16:26	V File	1 KB	
	BCD_tf.v	2019, 10, 01, 16:25	V File	2 KB	
	LOGSYS_SP3E.ucf	2019, 09, 19, 18:19	UCF File	6 KB	
×					
le nam	e: LOGSYS_SP3E.ucf		 Sources 	(*.txt *.vhd *.vhdl *.v	*.1

Open

Cancel

- The following window appears:
- Press OK
- If you have difficulties, ask for assistance



 Now open the previously added file. Uncomment the clk signal first: select the line, right click and select Uncomment-> Line(s)

```
# LOGSYS Fejlesztőkábel GUI vezérlő és ko
10
    #NET "mosi"
11
                   LOC = "P120";
    #NET "miso"
                    LOC = "P143";
12
    #NET "clk"
                       I \cap C = -
                                יי<u>פרו</u>חיי
13
                                            PULLDOWN;
                     🔄 Undo
                                Ctrl+Z
                                     9"
    #NET "rst"
                                           PULLDOWN;
14
                                Ctrl+Y
                     🖼 Redo
15
                                Ctrl+X
                       Cut
16
                                     mógomb, balról -
      4 darab ak
                                Ctrl+C
                       Copy
17
    #NET "bt<3>"
                       Paste
                                Ctrl+V
    #NET "bt<2>" X Delete
                                     ht –
18
                                Del
                                      1
    #NET "bt<1>"
19
                       Comment
                                   ٠
                       Uncomment
                                     🏪 Line(s)
20
    #NET "bt<0>"
                                             Alt+Shift+C
```

• Then uncomment the sw signals:



 Uncomment the seg_n signals, these are needed for the 7-segment display:



• Finally, uncomment the dig_n signals



- Now you can generate the programming file
- First, left click on the Top module in the top left corner

• Then right click on "Generate Programming" File", and select "Rerun All"



₽t

9:

ŧ٠

?:

• After the generation, you should see this.



• If you see errors or warnings, ask for assistance.

- If the program file was generated successfully, you can connect the FPGA board to the PC
- Mind the orientation of the JTAG connector!



• Launch the Logsys GUI application



• Press the +5V button to turn the board on

Power Voltage +5V On	Current Maximum Value: D Log to <u>file</u>	450 🗸 mA	
Measurement			
+5Vout: 4,92	V Maximum Value:	500 🚖 mA	
I/Oref: 3,31	V Critical Value:	90 🜲 %	
JTAGref: 2,50	V Samples/Second:	10 🜩	

• On the right side of the screen, select JTAG download:

		•	
	LDC221		џ
	Info LOGSYS development cable	Configuration 🖂 JTAG Download	
Press the	"Query JTAC	ິງ chain" bເ	utton
	JTAG	Devices in the JTA	i chain:
	Query JTAG chain	XC3S250E (Xilinx)	~
	Clear Log		

• Then press "Configure the Selected Device"

Devices in the JTAG chain:		
XC3S250E (Xilinx)	\sim	Configure the selected device

 Browse the generated file in your working directory

• Press Open



- The circuit requires to add a system clock input for the circuit
- First set the value of the clock frequency to 1000
- Press the Set button
- Click into the CLK checkbox
- Note: on Windows 10 the tick might not appear, but it should be fine



- Now try to set different inputs on the switches. Can you see the correct values and their sum on the 7-segment display?
- If you want to see the circuit in "slow motion", first set the frequency to 1-2 Hz, press then press the Set button.
- Now you can see the circuit selecting the individual digits one by one.
- This can't be seen when the clock frequency is high, due to the dynamics of the human eye.