

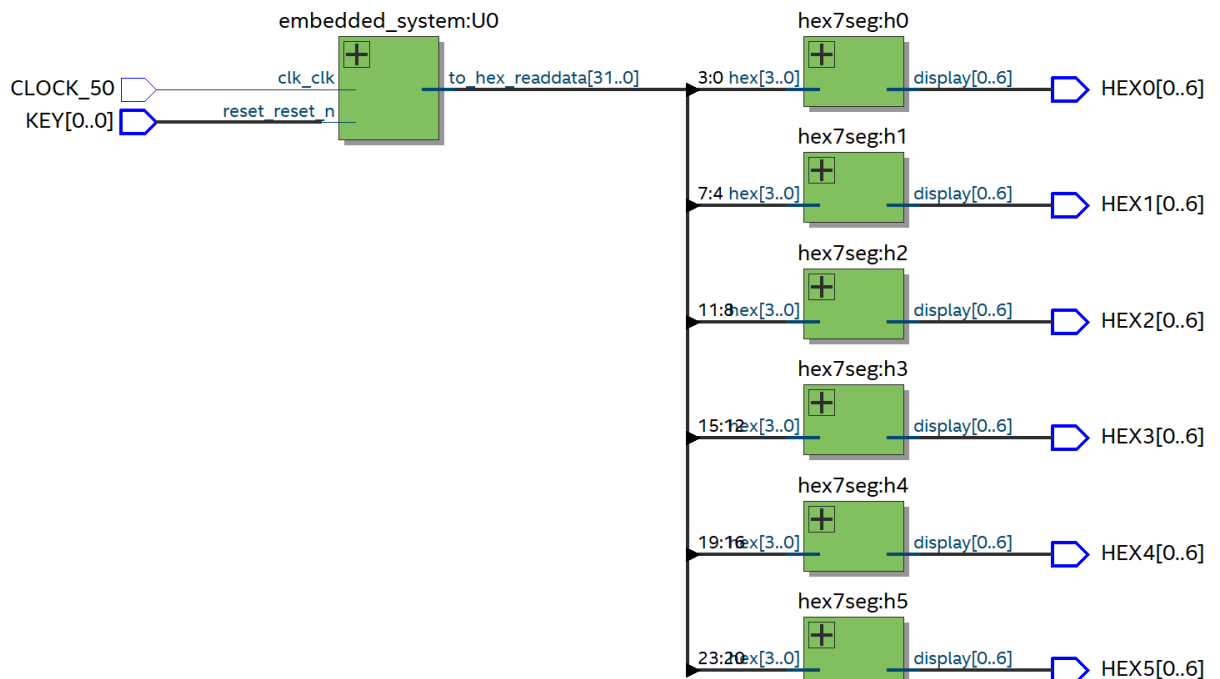
## Лабораторна робота №2

## Завдання 1

До проекту було додано HEX3, HEX4, HEX5, щоб були задіяні всі семисегментні індикатори.

```
module component_tutorial (CLOCK_50, KEY,
                           HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);

    input CLOCK_50;
    input [0:0] KEY;
    output [0:6] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
    wire [23:0] to_HEX;
    embedded_system U0 (
        .clk_clk(CLOCK_50),
        .reset_reset_n(KEY[0]),
        .to_hex_readdata(to_HEX)
    );
    hex7seg h0(to_HEX[3:0], HEX0);
    hex7seg h1(to_HEX[7:4], HEX1);
    hex7seg h2(to_HEX[11:8], HEX2);
    hex7seg h3(to_HEX[15:12], HEX3);
    hex7seg h4(to_HEX[19:16], HEX4);
    hex7seg h5(to_HEX[23:20], HEX5);
endmodule
```



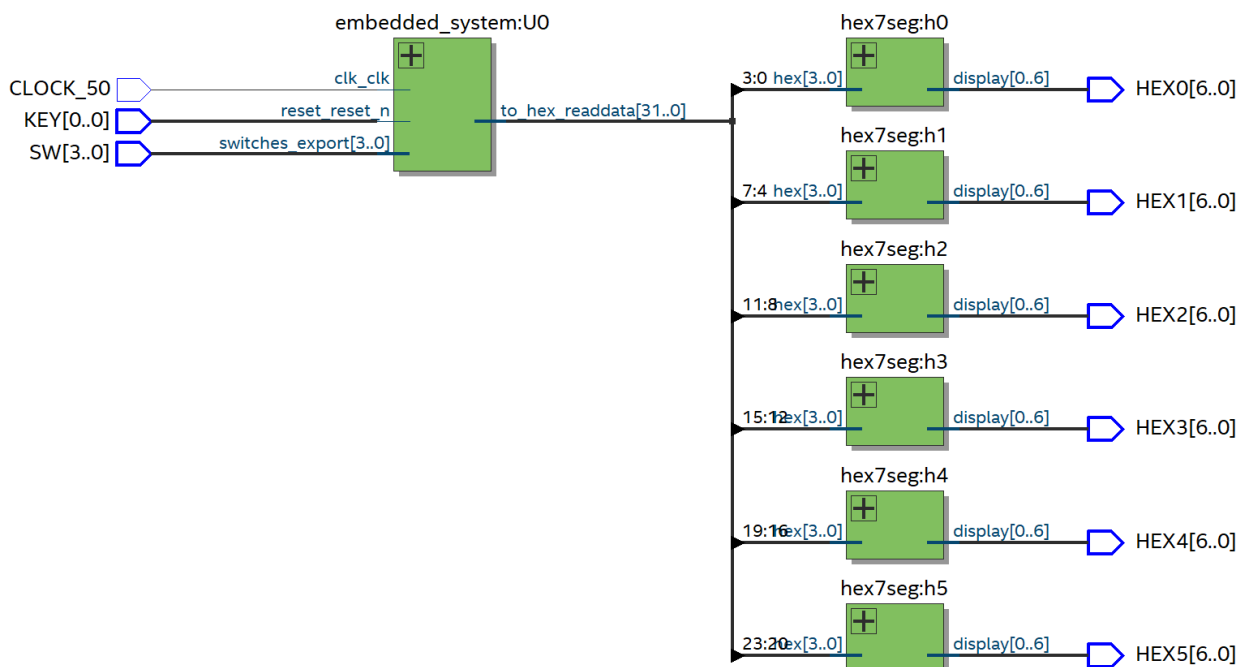
## Завдання 2

До проекту було додано перемикачі SW0 – SW3 для формування значення, яке повинно виводитись на семисегментні індикатори.

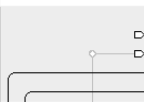
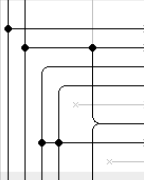

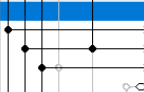
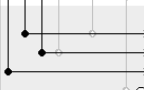
```
module component_tutorial (CLOCK_50, KEY, SW,
                           HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);

    input CLOCK_50;
    input [0:0] KEY;
    input [3:0] SW;
    output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
    wire [23:0] to_HEX;
    embedded_system U0 (
        .clk_clk(CLOCK_50),
        .reset_reset_n(KEY[0]),
        .switches_export(SW),
        .to_hex_readdata(to_HEX),
    );

    hex7seg h0(to_HEX[3:0], HEX0);
    hex7seg h1(to_HEX[7:4], HEX1);
    hex7seg h2(to_HEX[11:8], HEX2);
    hex7seg h3(to_HEX[15:12], HEX3);
    hex7seg h4(to_HEX[19:16], HEX4);
    hex7seg h5(to_HEX[23:20], HEX5);
endmodule
```



У Platform Designer було додано блок вводу/виводу switches для перемикачів.

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		<div>clk_0</div> <div>clk_in</div> <div>clk_in_reset</div> <div>clk</div> <div>clk_reset</div>	<div>Clock Source</div> <div>Clock Input</div> <div>Reset Input</div> <div>Clock Output</div> <div>Reset Output</div>	<div>clk</div> <div>reset</div> <div>Double-click to export</div> <div>Double-click to export</div>	exported clk_0			
<input checked="" type="checkbox"/>		<div>nios2_qsys_0</div> <div>clk</div> <div>reset_n</div> <div>data_master</div> <div>instruction_master</div> <div>avalon_memory_mapped_master</div> <div>d_irq</div> <div>interrupt_receiver</div> <div>reset_output</div> <div>jtag_debug_module_reset</div> <div>jtag_debug_module</div> <div>avalon_memory_mapped_slave</div> <div>custom_instruction_master</div>	<div>Nios II (Classic) Processor</div> <div>Clock Input</div> <div>Reset Input</div> <div>Avalon Memory Mapped Master</div> <div>Avalon Memory Mapped Master</div> <div>Interrupt Receiver</div> <div>Reset Output</div> <div>Custom Instruction Master</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	clk_0 [clk] [clk] [clk] [clk] [clk] [clk]	# 0x3000	0x37ff	IRQ 0 IRQ 31
<input checked="" type="checkbox"/>		<div>onchip_memory2_0</div> <div>clk1</div> <div>s1</div> <div>reset1</div>	<div>On-Chip Memory (RAM or ROM)...</div> <div>Clock Input</div> <div>Avalon Memory Mapped Slave</div> <div>Reset Input</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	clk_0 [clk1] [clk1]	# 0x2000	0x2fff	
<input checked="" type="checkbox"/>		switches	P10 (Parallel I/O) Intel FPGA IP	Double-click to export Double-click to export Double-click to export	clk_0 [clk] [clk]	# 0x0020	0x002f	
<input checked="" type="checkbox"/>		<div>reg32_avalon_interface_0</div> <div>reg32_component</div> <div>reset_input</div> <div>avalon_memory_mapped_slave</div> <div>clock_reset</div> <div>avalon_slave_0</div> <div>clock_sink</div> <div>conduit_end</div>	<div>reg32_component</div> <div>Reset Input</div> <div>Avalon Memory Mapped Slave</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	[clock_sink] [clock_sink] clk_0 [clock_sink]	# 0x0000	0x0003	

Программный код:

```
#define switches (volatile char *) 0x0000020
#define reg32 (char *) 0x0000000
void main()
{
    while (1)
        *reg32 = *switches;
}
```