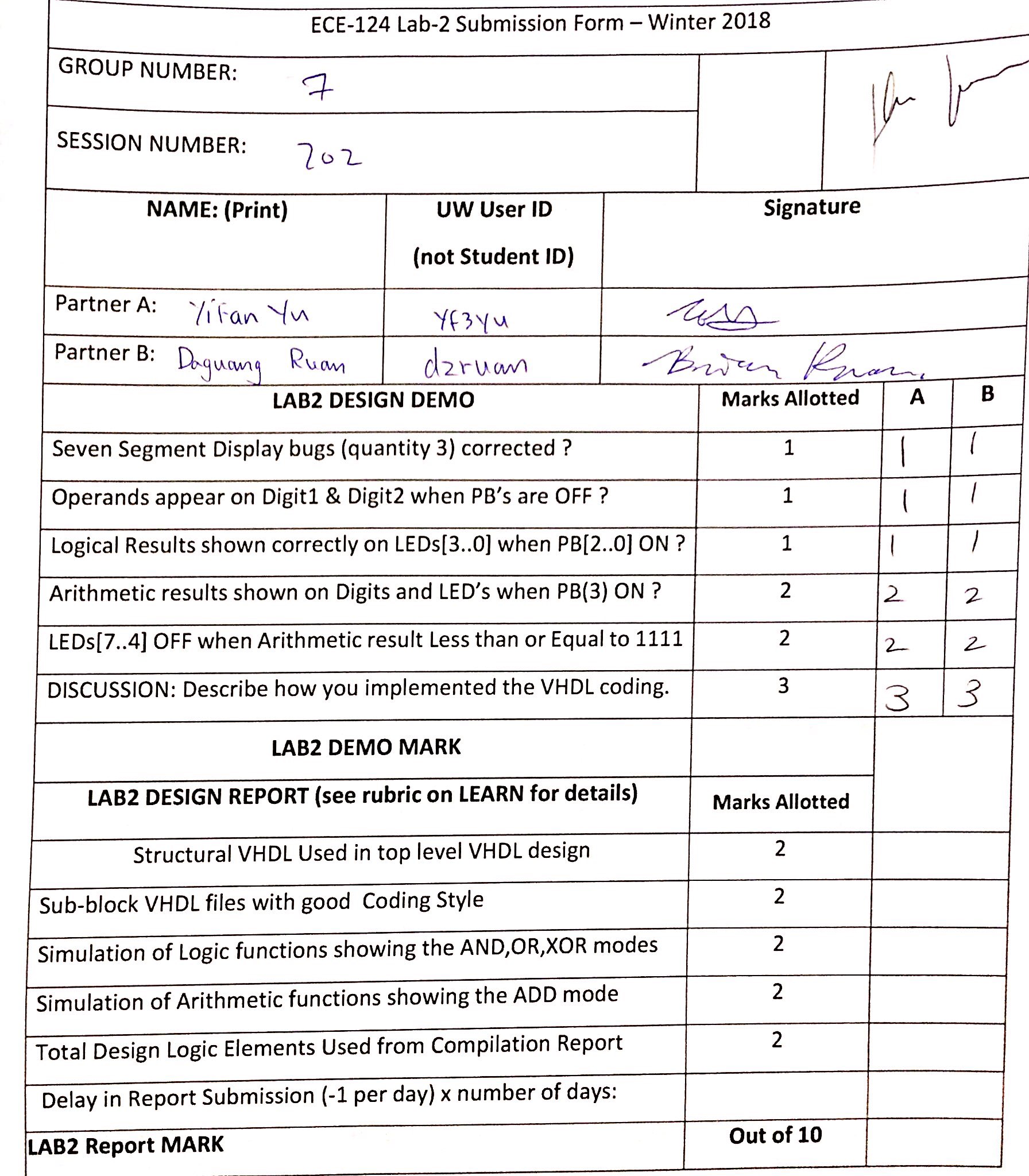
LAB2 GRP7 SESS202 REPORT



1. Top level VHDL FILE:

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity LogicalStep\_Lab2\_top is port (

clkin\_50 : in std\_logic;

pb : in std\_logic\_vector(3 downto 0);

sw : in std\_logic\_vector(7 downto 0); -- The switch inputs

leds : out std\_logic\_vector(7 downto 0); -- for displaying the switch content

seg7\_data : out std\_logic\_vector(6 downto 0); -- 7-bit outputs to a 7-segment

seg7\_char1 : out std\_logic; -- seg7 digit1 selector

seg7\_char2 : out std\_logic -- seg7 digit2 selector

);

end LogicalStep\_Lab2\_top;

architecture SimpleCircuit of LogicalStep\_Lab2\_top is

--

-- Components Used ---

-------------------------------------------------------------------

component SevenSegment port ( -- converts a 4-bit number to 7 bit-number to be use for the 7-segment display

hex : in std\_logic\_vector(3 downto 0); -- The 4 bit data to be displayed

sevenseg : out std\_logic\_vector(6 downto 0) -- 7-bit outputs to a 7-segment

);

end component;

component concatenate port ( -- concatenate 2 signals(4-bit numbers)

hexA : in std\_logic\_vector(3 downto 0);

hexB : in std\_logic\_vector(3 downto 0);

output : out std\_logic\_vector(7 downto 0)

);

end component;

component mux port ( --selector to let pass hexin1 or hexin2

hex\_in1, hex\_in2 : in std\_logic\_vector(7 downto 0);

mux\_select : in std\_logic;

hex\_out : out std\_logic\_vector(7 downto 0)

);

end component;

component segment7\_mux port (

clk : in std\_logic :='0';

DIN2 : in std\_logic\_vector(6 downto 0);

DIN1 : in std\_logic\_vector(6 downto 0);

DOUT : out std\_logic\_vector(6 downto 0);

DIG2 : out std\_logic;

DIG1 : out std\_logic

);

end component;

component add port( --function for adding two 4-bit numbers and outputs a 8-bit number

hexA : in std\_logic\_vector(3 downto 0);

hexB : in std\_logic\_vector(3 downto 0);

output : out std\_logic\_vector(7 downto 0)

);

end component;

component Logic\_Processor is port ( --function seletor between add, xor, or

hexA : in std\_logic\_vector(3 downto 0);

hexB : in std\_logic\_vector(3 downto 0);

pressButton : in std\_logic\_vector ( 2 downto 0);

output : out std\_logic\_vector(7 downto 0)

);

end component;

-- Create any signals, or temporary variables to be used

--

-- std\_logic\_vector is a signal which can be used for logic operations such as OR, AND, NOT, XOR

--

signal seg7\_A : std\_logic\_vector(6 downto 0); -- final output for hexA to seg7

signal hex\_A : std\_logic\_vector(3 downto 0); -- input number 1

signal seg7\_B : std\_logic\_vector(6 downto 0); --final output for hexB to seg7

signal hex\_B : std\_logic\_vector(3 downto 0); --input number 2

signal concatenationResult : std\_logic\_vector(7 downto 0); --result after concatenation of both inputs

signal sumResult: std\_logic\_vector(7 downto 0); --result after adition of the two inputs

signal arithmetic\_result : std\_logic\_vector(7 downto 0); -- = sum if pb[3] is pressed ELSE = concatenation result

signal logicOutput: std\_logic\_vector (7 downto 0); ----result obtain after the logical processor

-- Here the circuit begins

begin

hex\_A <= sw(3 downto 0); --assign input 1 to switches 3 to 0

hex\_B <= sw(7 downto 4); --assign input 2 to switches 7 to 4

INST1: SevenSegment port map(arithmetic\_Result(7 downto 4), seg7\_A); --ports input 2 to 7-segment display on the fpga

INST2: SevenSegment port map(arithmetic\_Result(3 downto 0), seg7\_B); --ports input 1 to 7-segment display on the fpga

INST3: segment7\_mux port map(clkin\_50, seg7\_B, seg7\_A, seg7\_data, seg7\_char2, seg7\_char1); --

INST4: concatenate port map( hex\_B, hex\_A, concatenationResult); -- ports concatenated inputs

INST5: add port map( hex\_B, hex\_A, sumResult); --ports added result

INST6: mux port map(concatenationResult, sumResult, not pb(3), arithmetic\_Result); --ports concatenationResult

INST7: Logic\_Processor port map(hex\_A, hex\_B, pb (2 downto 0), logicOutput); -- reverted at the lower level

INST8: mux port map(sumResult, logicOutput, pb(3), leds); --ports mux result depending on selected buttons

end SimpleCircuit;

1. Subordinate VHDL files

* Mux.vhdl

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity mux is port (

hex\_in1, hex\_in2 : in std\_logic\_vector(7 downto 0); -- input 1 and 2

mux\_select :in std\_logic; -- selector

hex\_out : out std\_logic\_vector(7 downto 0)

);

end mux;

architecture mux\_logic of mux is

begin

hex\_out <= hex\_in1 when (mux\_select = '0') else hex\_in2;

-- output gets input 1 if selector is 0 (button is pushed) else get input2

end mux\_logic;

* Add.vhdl

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity add is port (

hexA : in std\_logic\_vector(3 downto 0); --input 1

hexB : in std\_logic\_vector(3 downto 0); --input 2

output : out std\_logic\_vector(7 downto 0) --output for added result

);

end add;

architecture Behavioral of add is

--

begin

--trying to add hexA to hexB

output(7 downto 0) <=std\_logic\_vector(unsigned("0000" & hexA) + unsigned("0000" & hexB)); --function to add both inputs(concatenated)

end architecture Behavioral;

----------------------------------------------------------------------

* Concatenate.vhdl

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity concatenate is port (

hexA : in std\_logic\_vector(3 downto 0); --input 1

hexB : in std\_logic\_vector(3 downto 0); --input 2

output : out std\_logic\_vector(7 downto 0) --concatenate result

);

end concatenate;

architecture Behavioral of concatenate is

--

begin

output <= hexA & hexB; --function for concatenation

end architecture Behavioral;

-----------------------------------------------------------------------------------------

* SevenSegment.vhdl

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

-------------------------------------------------------------------------

-- 7-segment display driver. It displays a 4-bit number on a 7-segment

-- This is created as an entity so that it can be reused many times easily

--

entity SevenSegment is port (

hex : in std\_logic\_vector(3 downto 0); -- The 4 bit data to be displayed

sevenseg : out std\_logic\_vector(6 downto 0) -- 7-bit outputs to a 7-segment

);

end SevenSegment;

architecture Behavioral of SevenSegment is

--

-- The following statements convert a 4-bit input, called dataIn to a pattern of 7 bits

-- The segment turns on when it is '1' otherwise '0'

--

begin

with hex select --GFEDCBA 3210 -- data in

sevenseg <= "0111111" when "0000", -- [0]

"0000110" when "0001", -- [1]

"1011011" when "0010", -- [2] +---- a -----+

"1001111" when "0011", -- [3] | |

"1100110" when "0100", -- [4] | |

"1101101" when "0101", -- [5] f b

"1111101" when "0110", -- [6] | |

"0000111" when "0111", -- [7] | |

"1111111" when "1000", -- [8] +---- g -----+

"1101111" when "1001", -- [9] | |

"1110111" when "1010", -- [A] | |

"1111100" when "1011", -- [b] e c

"1011000" when "1100", -- [c] | |

"1011110" when "1101", -- [d] | |

"1111001" when "1110", -- [E] +---- d -----+

"1110001" when "1111", -- [F]

"0000000" when others; -- [ ]

end architecture Behavioral;

----------------------------------------------------------------------

* Logic\_Processor.vhdl

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity Logic\_Processor is port (

hexA : in std\_logic\_vector(3 downto 0); --input 1

hexB : in std\_logic\_vector(3 downto 0); --input 2

pressButton : in std\_logic\_vector ( 2 downto 0); --buttons for operator selection

output : out std\_logic\_vector(7 downto 0) --output of the operation

);

end Logic\_Processor;

architecture Behavioral of Logic\_Processor is

signal operator : std\_logic\_vector(2 downto 0);

begin

operator <= not pressButton(2) & not pressButton(1) & not pressButton(0);

with operator select

output<= "0000" & (hexA and hexB) when "001", --define the operator depending on button inputs

"0000" & (hexA or hexB) when "010",

"0000" & (hexA xor hexB) when "100",

"00000000" when others;

end architecture Behavioral;

----------------------------------------------------------------------

1. Supporting Images

Image1: graphical simulation of the add functionality

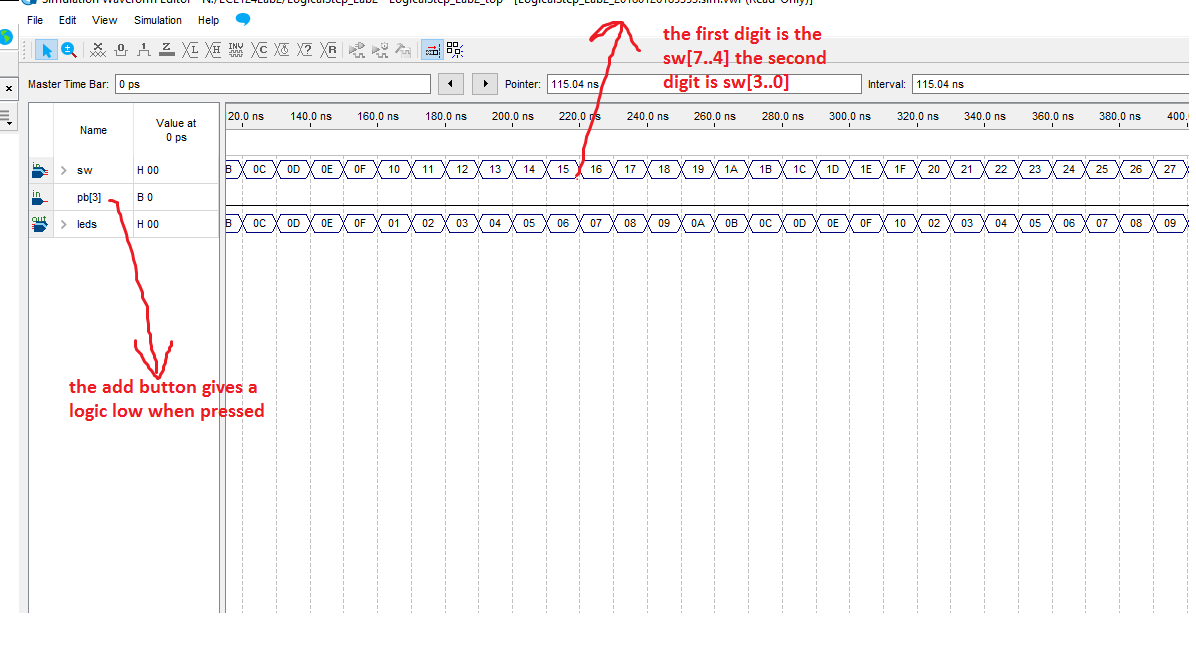


Image2: graphical simulation of the logic gates (AND, XOR, OR) functionalities

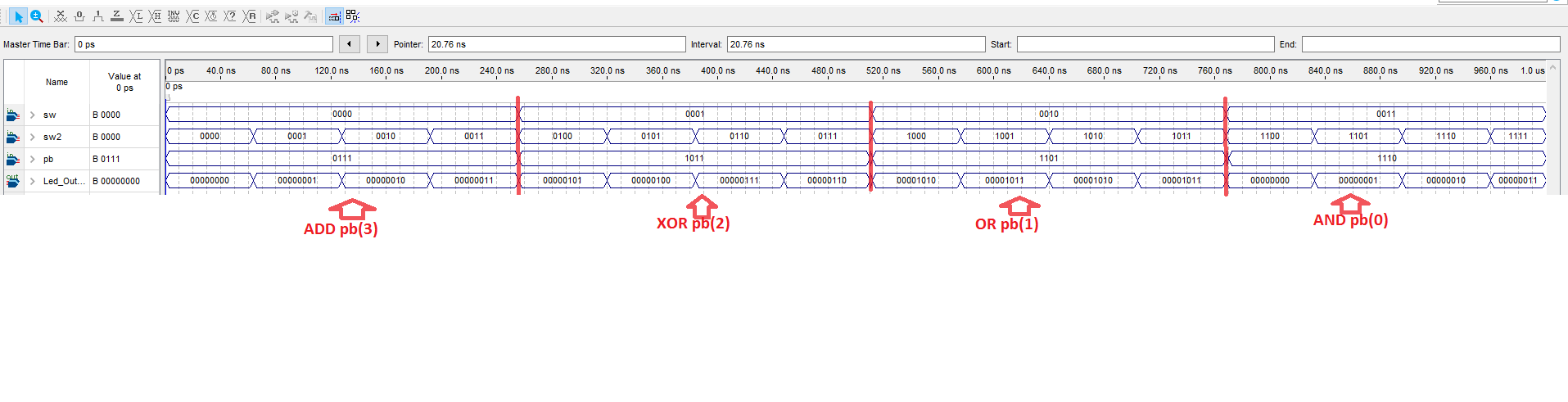


Image3: RTL diagram of all the gates representing a crude ALU

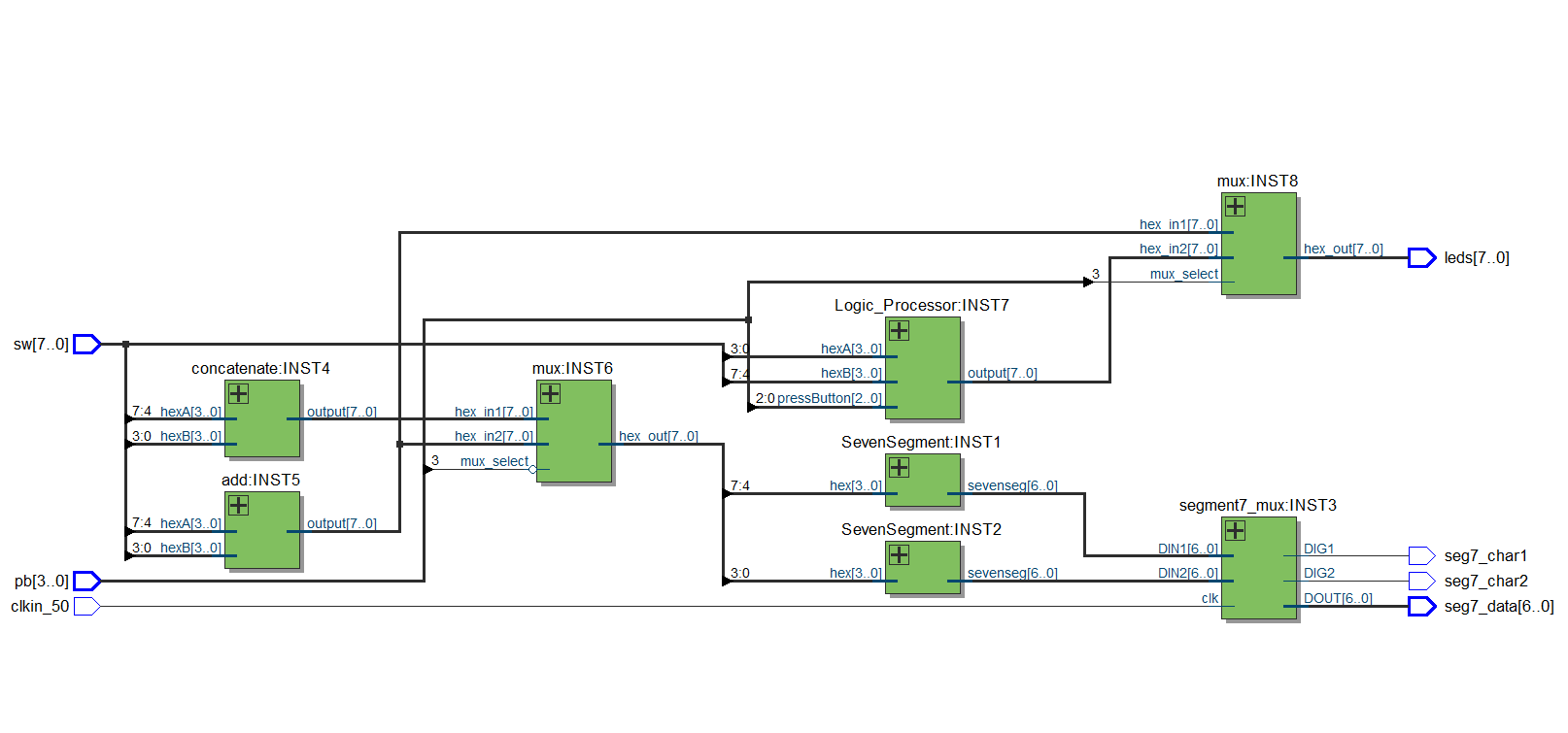


Image 4: Compilation log on Quartus Prime and total number of logic elements

