# VHDL Files (Code)

## LogicalStep\_Lab3\_top

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity LogicalStep\_Lab3\_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 digit selectors

seg7\_char2 : out std\_logic -- seg7 digit selectors

);

end LogicalStep\_Lab3\_top;

architecture Energy\_Monitor of LogicalStep\_Lab3\_top is

-- Components Used--

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

component compx4 port(

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

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

A\_greater\_B : out std\_logic;

A\_equals\_B : out std\_logic;

A\_lesser\_B : out std\_logic

);

end component;

component EMCL port (

--Inputs

greaterinput : in std\_logic;

equalinput : in std\_logic;

lesserinput : in std\_logic;

FWBpb : in std\_logic\_vector (2 downto 0); --(pb2-Fdoor), (pb1-window), (pb0-bdoor)

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

);

end component;

component SevenSegment 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 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 ;

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

signal Current\_Temp : std\_logic\_vector(3 downto 0);

signal Desired\_Temp : std\_logic\_vector(3 downto 0);

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

signal greater : std\_logic;

signal equals : std\_logic;

signal lesser : std\_logic;

signal SystemLights : std\_logic\_vector(6 downto 0);

signal seg7\_A : std\_logic\_vector(6 downto 0);

signal seg7\_B : std\_logic\_vector(6 downto 0);

signal Temp\_Display : std\_logic\_vector(6 downto 0);

-- Here the circuit begins--

begin

Desired\_Temp <= sw(7 downto 4); -- A

Current\_Temp <= sw(3 downto 0); -- B

DoorsWindowsOpen <= NOT(pb(2 downto 0));

--Component Hookup—

INST1: compx4 port map (Desired\_Temp, Current\_Temp, greater, equals, lesser);

INST2: EMCL port map (greater, equals, lesser, DoorsWindowsOpen, leds);

INST3: SevenSegment port map (Desired\_Temp, seg7\_A); --Digit 2

INST4: SevenSegment port map (Current\_Temp, seg7\_B); --Digit 1

INST5: segment7\_mux port map (clkin\_50, seg7\_A, seg7\_B, seg7\_data, seg7\_char1, seg7\_char2);

end Energy\_Monitor;

# Subordinate Files

## Compx4

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity compx4 is port(

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

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

A\_greater\_B : out std\_logic;

A\_equals\_B : out std\_logic;

A\_lesser\_B : out std\_logic

);

end entity compx4;

architecture Comp4 of compx4 is

--COMPONENT USED--

component Compx1 port(

A : in std\_logic;

B : in std\_logic;

greater : out std\_logic;

equals : out std\_logic;

lesser : out std\_logic

);

end component;

--SIGNAL CREATION—

signal greater3: std\_logic;

signal equals3: std\_logic;

signal lesser3 : std\_logic;

signal greater2: std\_logic;

signal equals2: std\_logic;

signal lesser2 : std\_logic;

signal greater1: std\_logic;

signal equals1: std\_logic;

signal lesser1: std\_logic;

signal greater0: std\_logic;

signal equals0: std\_logic;

signal lesser0: std\_logic;

begin

INST1: Compx1 port map (Operand\_A(3), Operand\_B(3),greater3,equals3,lesser3);

INST2: Compx1 port map (Operand\_A(2), Operand\_B(2),greater2,equals2,lesser2);

INST3: Compx1 port map (Operand\_A(1), Operand\_B(1),greater1,equals1,lesser1);

INST4: Compx1 port map (Operand\_A(0), Operand\_B(0),greater0,equals0,lesser0);

A\_greater\_B <= greater3 OR (equals3 AND greater2) OR (equals3 AND equals2 AND greater1) OR (equals3 AND equals2 AND equals1 AND greater0);

A\_lesser\_B <= lesser3 OR (equals3 AND lesser2) OR (equals3 AND equals2 AND lesser1) OR (equals3 AND equals2 AND equals1 AND lesser0);

A\_equals\_B <= equals3 AND equals2 AND equals1 AND equals0;

end Comp4;

## Compx1

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity Compx1 is port (

A : in std\_logic;

B : in std\_logic;

greater : out std\_logic;

equals : out std\_logic;

lesser : out std\_logic

);

end entity Compx1;

architecture Comp of Compx1 is

begin

greater <= A AND NOT(B);

equals <= A XNOR B;

lesser <= NOT(A) AND B;

end Comp;

## EMCL

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity EMCL is port (

--Inputs--

greaterinput : in std\_logic;

equalinput : in std\_logic;

lesserinput : in std\_logic;

FWBpb : in std\_logic\_vector(2 downto 0); --(pb2-Fdoor), (pb1-window), (pb0-bdoor)

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

*--Independant Outputs (Comments for Understanding)*

*--FWBpb : std\_logic\_vector (2 downto 0);*

*--Furnace\_ON : std\_logic;*

*--At\_Temp : std\_logic;*

*--AC\_ON : std\_logic;*

*--Dependant Outputs*

*--Blower\_ON : std\_logic;*

*--FrontDoor\_OPEN : std\_logic;*

*--Windows\_OPEN : std\_logic;*

*--BackDoor\_OPEN : std\_logic*

);

end entity EMCL;

architecture EMCL of EMCL is

begin

led(4) <= FWBpb(0); --BackDoor\_OPEN

led(5) <= FWBpb(1); --Windows\_OPEN

led(6) <= FWBpb(2); --FrontDoor\_OPEN

led(0) <= greaterinput AND (NOT(FWBpb(0)) AND NOT(FWBpb(1)) AND NOT(FWBpb(2))); --FURNACE\_ON

led(1) <= equalinput; -- AT\_TEMP

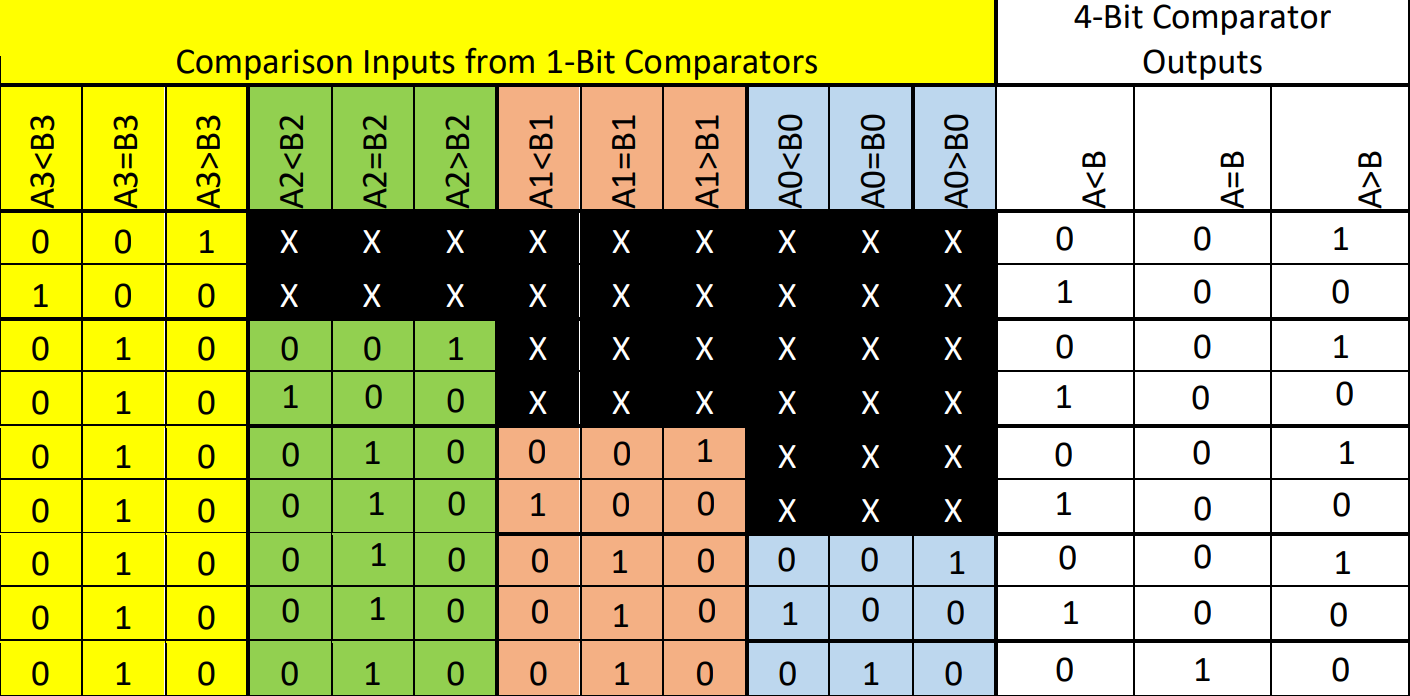
led(2) <= lesserinput AND (NOT(FWBpb(0)) AND NOT(FWBpb(1)) AND NOT(FWBpb(2)));

-- AC\_ON

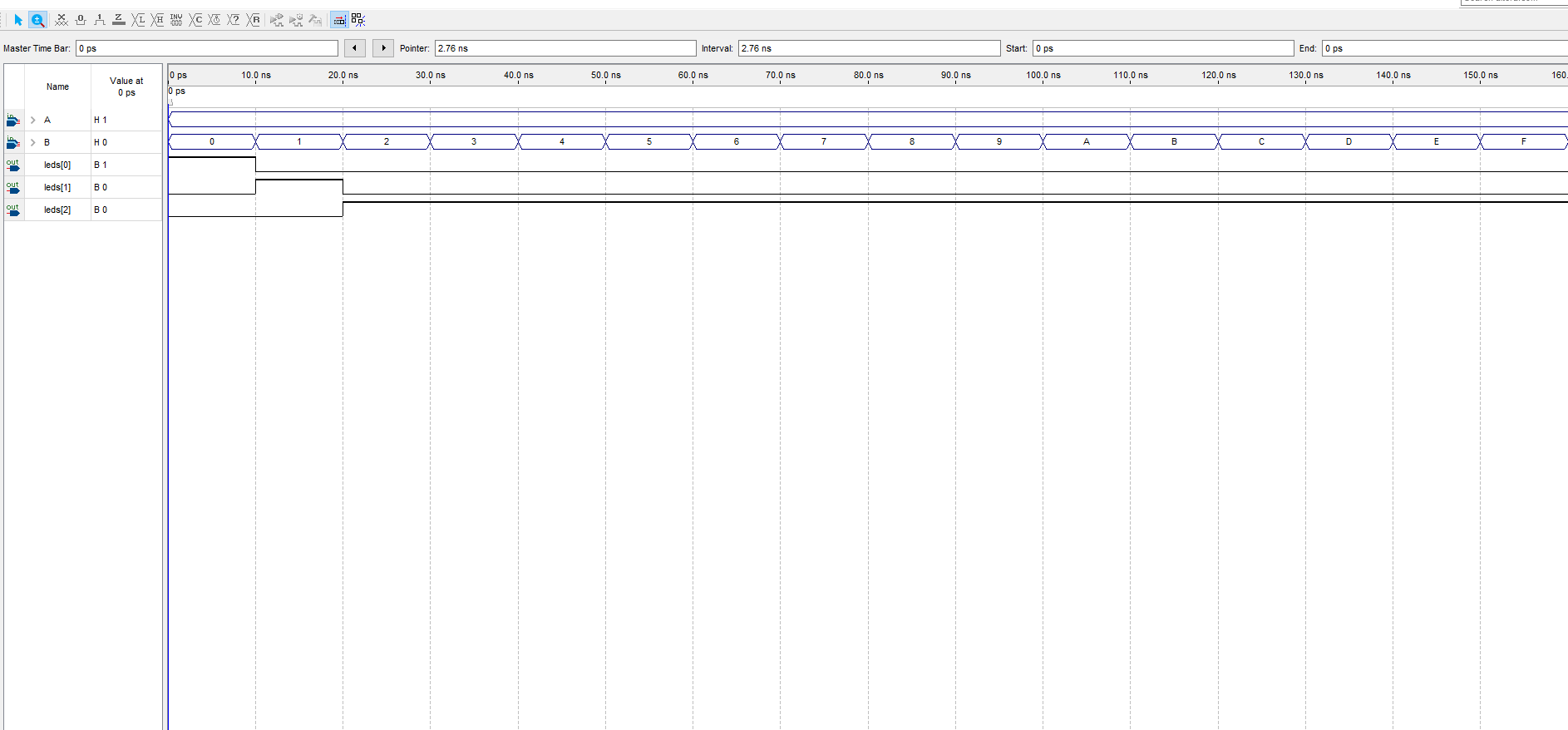
led(3) <= (lesserinput OR greaterinput) AND NOT(FWBpb(0)) AND NOT(FWBpb(1)) AND NOT(FWBpb(2)); --BLOWER\_ON

end EMCL;

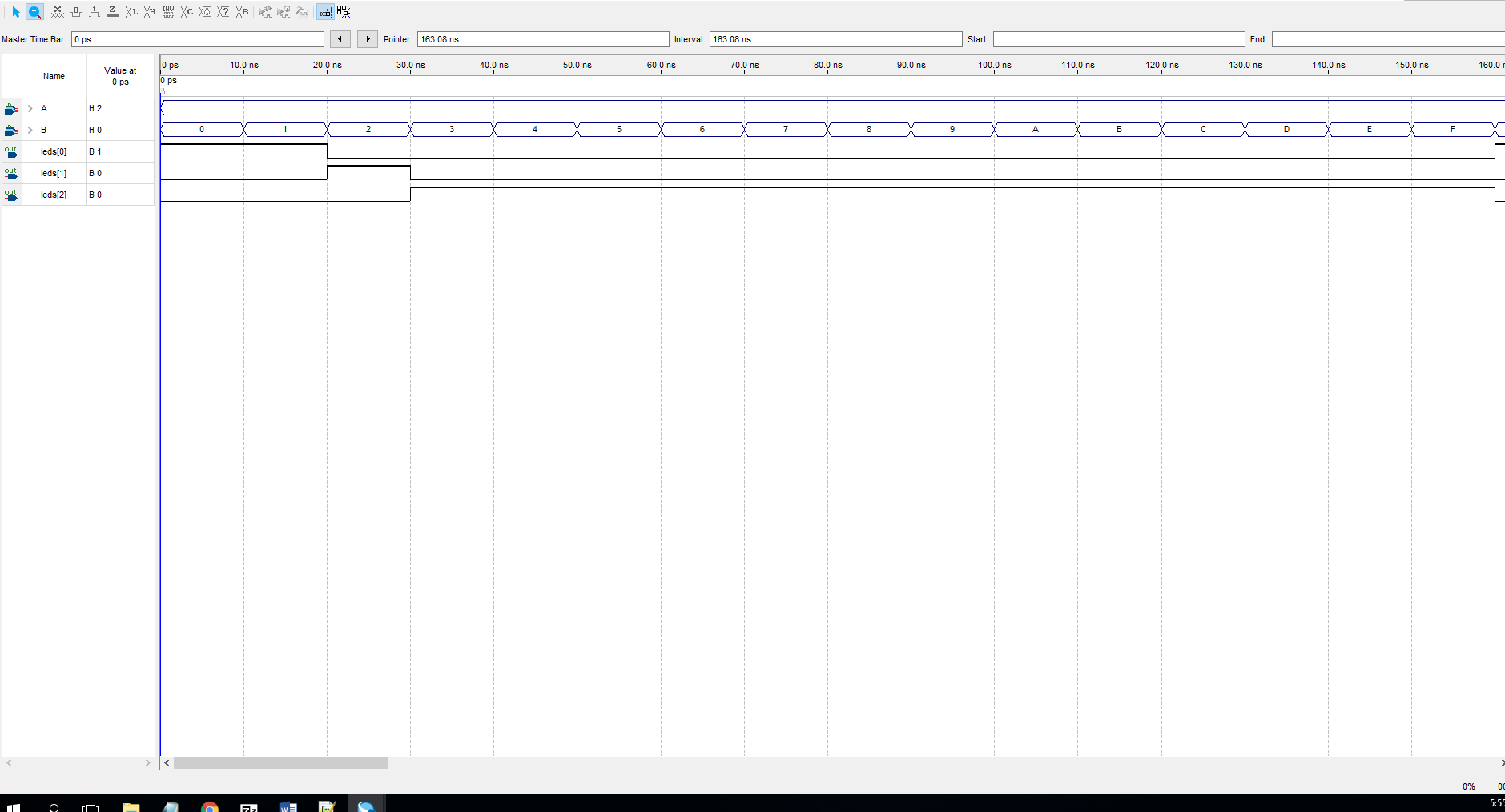
Truth Table



Simulations

Simulation 1:

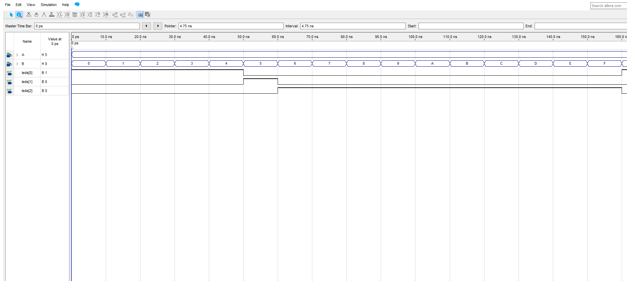
Input\_A set to a static value of hex 1. Input\_B run through all hex values between hex 0 and hex F.

Simulation 2:

Input\_A set to a static value of hex 2. Input\_B run through all hex values between hex 0 and hex F.

### Simulation 3:

Input\_A set to a static value of hex 8. Input\_B run through all hex values between hex 0 and hex F.

Simulation 4:

Input\_A set to a static value of hex 5. Input\_B run through all hex values between hex 0 and hex F.

RTL View of Logic Design

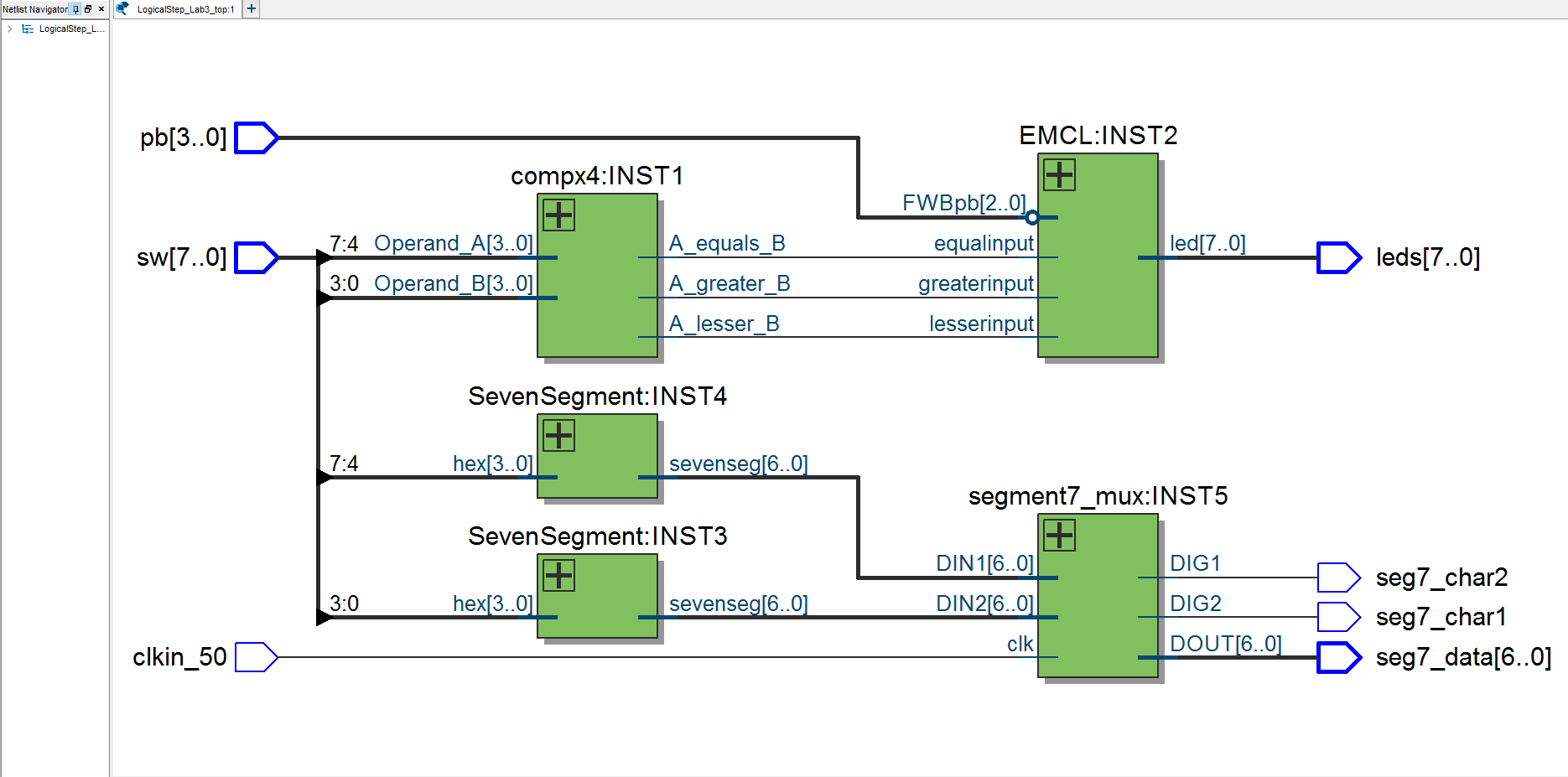
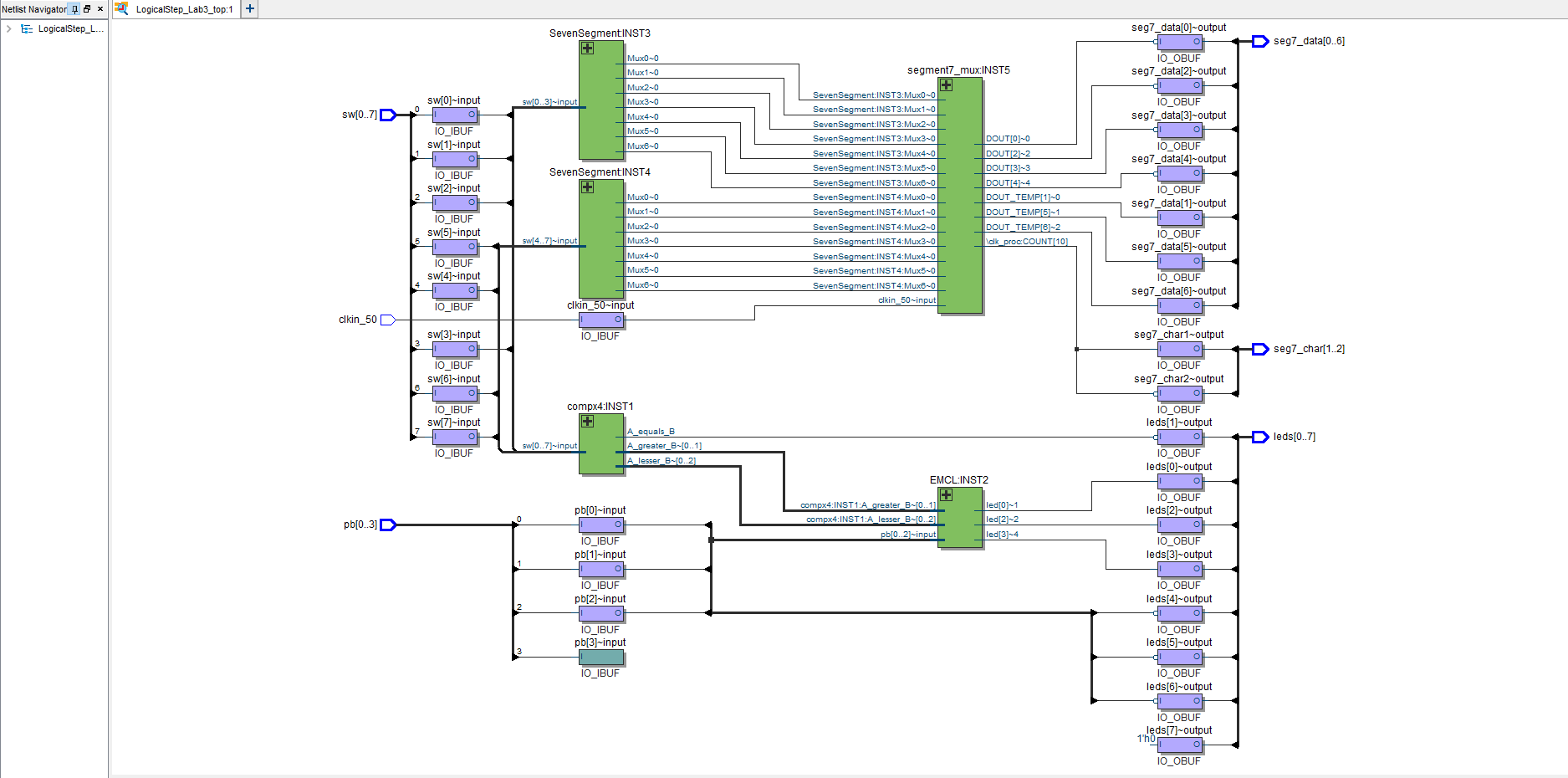
Figure 1 RTL View ­­

Figure 2 Technology Mapping Post-Mapping

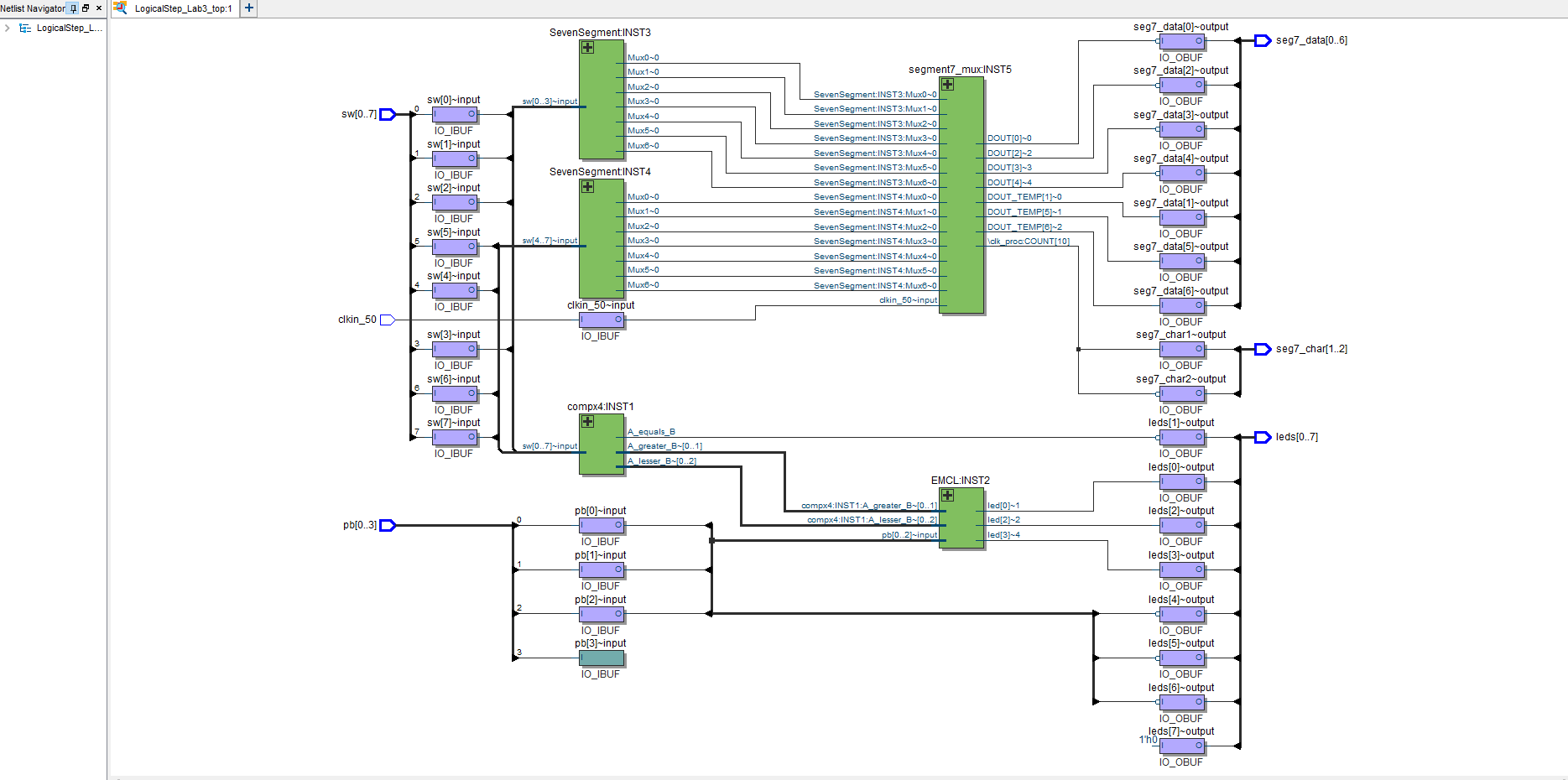
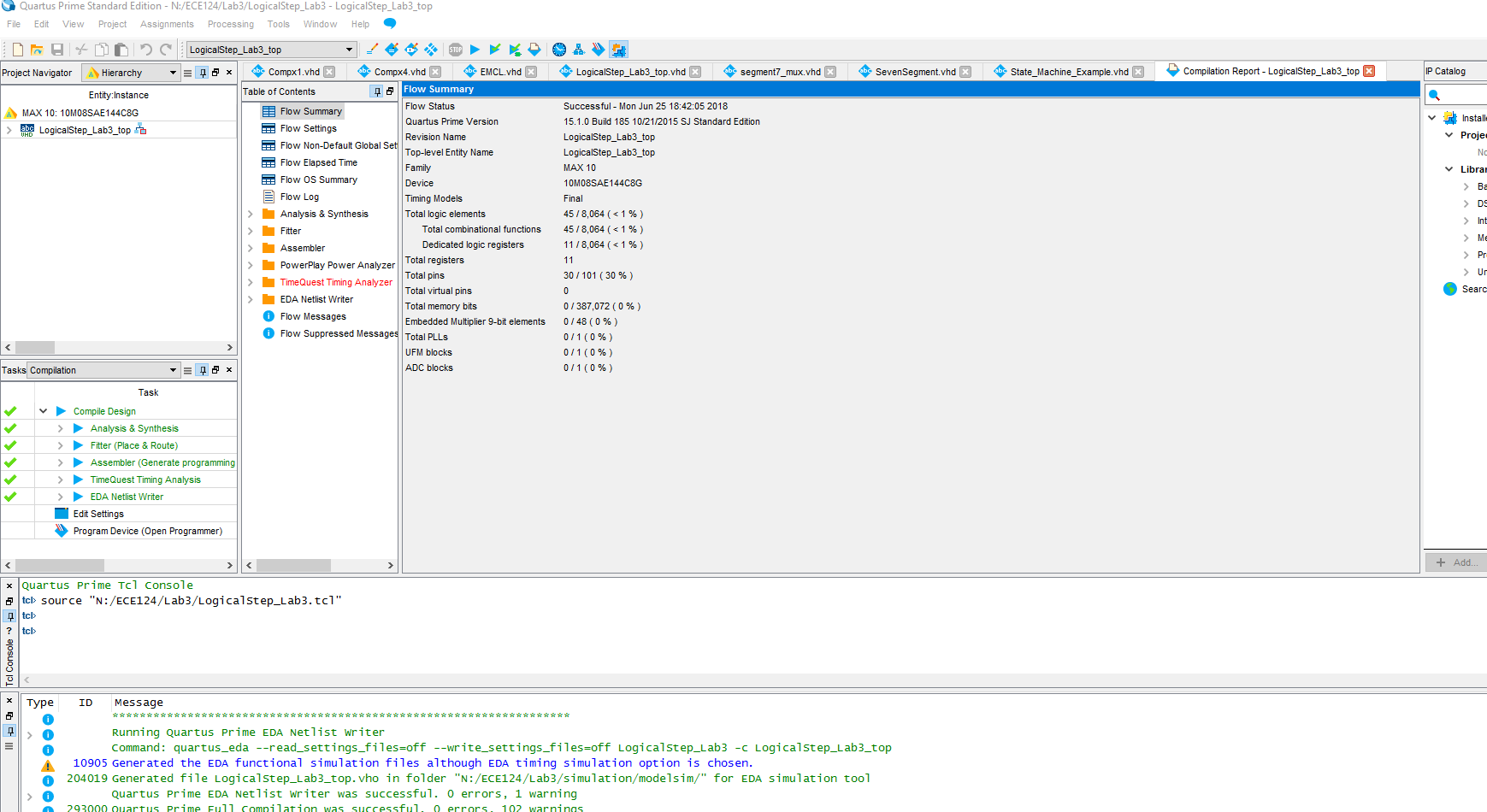


Figure 3 Technology Mapping Post-Fitting

Total Logical Elements

The total number of design logic elements are **45**.