Main Module Code

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

Library UNISIM;

use UNISIM.VComponents.all;

entity MorseCodeTranslatorMain is

Port ( SwitchBus : in STD\_LOGIC\_VECTOR(5 downto 0);

Clock : in STD\_LOGIC;

New\_Letter : in STD\_LOGIC;

Print\_Word : in STD\_LOGIC;

MorseCode : out STD\_LOGIC\_VECTOR(13 downto 0);

SigLED : out STD\_LOGIC;

Anode : out STD\_LOGIC\_VECTOR (3 downto 0);

Cathode : out STD\_LOGIC\_VECTOR (0 to 6);

SigPrint : out STD\_LOGIC);

type Morse is array (0 to 28) of STD\_LOGIC\_VECTOR(24 downto 0);

end MorseCodeTranslatorMain;

architecture Behavioral of MorseCodeTranslatorMain is

constant OnOff : integer := (40000000);

signal letters : character;

signal MasterList : STD\_LOGIC\_VECTOR(24 downto 0);

constant MAX : integer := (200000);

signal A : STD\_LOGIC\_VECTOR(5 downto 0) := SwitchBus(5 downto 0);

signal B : STD\_LOGIC\_VECTOR(5 downto 0) := SwitchBus(5 downto 0);

begin

A <= SwitchBus(5 downto 0);

B <= SwitchBus(5 downto 0);

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

Segment\_Display : process(Clock) is

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

variable digit : STD\_LOGIC := '0';

variable CLK : integer := 0;

begin

if rising\_edge(Clock) then

if (CLK = MAX) then

case digit is

when '0' => Anode <= "1110";

case A is -- ABCDEFG

when "000000" => Cathode <= "0000001"; -- 0

when "000001" => Cathode <= "1001111"; -- 1

when "000010" => Cathode <= "0010010"; -- 2

when "000011" => Cathode <= "0000110"; -- 3

when "000100" => Cathode <= "1001100"; -- 4

when "000101" => Cathode <= "0100100"; -- 5

when "000110" => Cathode <= "0100000"; -- 6

when "000111" => Cathode <= "0001111"; -- 7

when "001000" => Cathode <= "0000000"; -- 8

when "001001" => Cathode <= "0001100"; -- 9

when "001010" => Cathode <= "0000001"; -- 10

when "001011" => Cathode <= "1001111"; -- 11

when "001100" => Cathode <= "0010010"; -- 12

when "001101" => Cathode <= "0000110"; -- 13

when "001110" => Cathode <= "1001100"; -- 14

when "001111" => Cathode <= "0100100"; -- 15

when "010000" => Cathode <= "0100000"; -- 16

when "010001" => Cathode <= "0001111"; -- 17

when "010010" => Cathode <= "0000000"; -- 18

when "010011" => Cathode <= "0001100"; -- 19

when "010100" => Cathode <= "0000001"; -- 20

when "010101" => Cathode <= "1001111"; -- 21

when "010110" => Cathode <= "0010010"; -- 22

when "010111" => Cathode <= "0000110"; -- 23

when "011000" => Cathode <= "1001100"; -- 24

when "011001" => Cathode <= "0100100"; -- 25

when "011010" => Cathode <= "0100000"; -- 26

when "011011" => Cathode <= "0001111"; -- 27

when "011100" => Cathode <= "0000000"; -- 28

when others => Cathode <= "1111111";

end case;

when '1' => Anode <= "1101";

case B is

when "000000" => Cathode <= "0000001"; -- 0

when "000001" => Cathode <= "0000001"; -- 1

when "000010" => Cathode <= "0000001"; -- 2

when "000011" => Cathode <= "0000001"; -- 3

when "000100" => Cathode <= "0000001"; -- 4

when "000101" => Cathode <= "0000001"; -- 5

when "000110" => Cathode <= "0000001"; -- 6

when "000111" => Cathode <= "0000001"; -- 7

when "001000" => Cathode <= "0000001"; -- 8

when "001001" => Cathode <= "0000001"; -- 9

when "001010" => Cathode <= "1001111"; -- 10

when "001011" => Cathode <= "1001111"; -- 11

when "001100" => Cathode <= "1001111"; -- 12

when "001101" => Cathode <= "1001111"; -- 13

when "001110" => Cathode <= "1001111"; -- 14

when "001111" => Cathode <= "1001111"; -- 15

when "010000" => Cathode <= "1001111"; -- 16

when "010001" => Cathode <= "1001111"; -- 17

when "010010" => Cathode <= "1001111"; -- 18

when "010011" => Cathode <= "1001111"; -- 19

when "010100" => Cathode <= "0010010"; -- 20

when "010101" => Cathode <= "0010010"; -- 21

when "010110" => Cathode <= "0010010"; -- 22

when "010111" => Cathode <= "0010010"; -- 23

when "011000" => Cathode <= "0010010"; -- 24

when "011001" => Cathode <= "0010010"; -- 25

when "011010" => Cathode <= "0010010"; -- 26

when "011011" => Cathode <= "0010010"; -- 27

when "011100" => Cathode <= "0010010"; -- 28

when others => Cathode <= "1111111";

end case;

when others => Cathode <= "1111111";

end case;

digit := not(digit);

CLK := 0;

else

CLK := integer(CLK + 1);

end if;

end if;

end process Segment\_Display;

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

Clock\_Counter : process (Clock, MasterList) is

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

variable num : integer := 0;

variable count : integer := 0;

variable holder : STD\_LOGIC;

begin

if rising\_edge(Clock) then

if MasterList(24 - num) = '0' then

if (count = OnOff) then

MorseCode(0) <= '0'; MorseCode(1) <= '0'; MorseCode(2) <= '0'; MorseCode(3) <= '0'; MorseCode(4) <= '0'; MorseCode(5) <= '0'; MorseCode(6) <= '0';

MorseCode(7) <= '0'; MorseCode(8) <= '0'; MorseCode(9) <= '0'; MorseCode(10) <= '0'; MorseCode(11) <= '0'; MorseCode(12) <= '0'; MorseCode(13) <= '0';

count := 0;

num := (num + 1);

else count := (count + 1);

end if;

elsif MasterList(24 - num) = '1' then

if (count = OnOff) then

MorseCode(0) <= '1'; MorseCode(1) <= '1'; MorseCode(2) <= '1'; MorseCode(3) <= '1'; MorseCode(4) <= '1'; MorseCode(5) <= '1'; MorseCode(6) <= '1';

MorseCode(7) <= '1'; MorseCode(8) <= '1'; MorseCode(9) <= '1'; MorseCode(10) <= '1'; MorseCode(11) <= '1'; MorseCode(12) <= '1'; MorseCode(13) <= '1';

count := 0;

num := (num + 1);

else count := (count + 1);

end if;

end if;

end if;

end process Clock\_Counter;

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

MakingStatement : Process (SwitchBus, Clock, New\_Letter, Print\_Word) is

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

begin

SigLED <= '0';

if new\_letter = '1' then

SigLED <= '1';

if(SwitchBus = "000000") then letters <= '\_';

elsif(SwitchBus = "000001") then letters <= 'A';

elsif(SwitchBus = "000010") then letters <= 'B';

elsif(SwitchBus = "000011") then letters <= 'C';

elsif(SwitchBus = "000100") then letters <= 'D';

elsif(SwitchBus = "000101") then letters <= 'E';

elsif(SwitchBus = "000110") then letters <= 'F';

elsif(SwitchBus = "000111") then letters <= 'G';

elsif(SwitchBus = "001000") then letters <= 'H';

elsif(SwitchBus = "001001") then letters <= 'I';

elsif(SwitchBus = "001010") then letters <= 'J';

elsif(SwitchBus = "001011") then letters <= 'K';

elsif(SwitchBus = "001100") then letters <= 'L';

elsif(SwitchBus = "001101") then letters <= 'M';

elsif(SwitchBus = "001110") then letters <= 'N';

elsif(SwitchBus = "001111") then letters <= 'O';

elsif(SwitchBus = "010000") then letters <= 'P';

elsif(SwitchBus = "010001") then letters <= 'Q';

elsif(SwitchBus = "010010") then letters <= 'R';

elsif(SwitchBus = "010011") then letters <= 'S';

elsif(SwitchBus = "010100") then letters <= 'T';

elsif(SwitchBus = "010101") then letters <= 'U';

elsif(SwitchBus = "010110") then letters <= 'V';

elsif(SwitchBus = "010111") then letters <= 'W';

elsif(SwitchBus = "011000") then letters <= 'X';

elsif(SwitchBus = "011001") then letters <= 'Y';

elsif(SwitchBus = "011010") then letters <= 'Z';

elsif(SwitchBus = "011011") then letters <= '.';

elsif(SwitchBus = "011100") then letters <= ',';

end if;

end if;

end process MakingStatement;

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

Translator : Process (Print\_Word, letters) is

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

variable MorseArray : Morse;

begin

SigPrint <= '0';

-- DOT = 01 DASH = 0111 Space Bettween Letters 00000 Ex. A = "01 0111 00000"

MorseArray := ( "010111", "0111010101", "011101011101", -- A B C

"01110101", "01", "01010111", --D E F

"0111011101", "01010101", "0101", -- G H I

"01011101110111", "0111010111", "0101110101", -- J K L

"01110111", "011101", "011101110111", -- M N O

"0101110111", "01110111010111", "01011101", -- P Q R

"010101", "011100000", "01010111", -- S T U

"0101010111", "0101110111", "011101010111", -- V W X

"01110101110111", "011101110101", "01010101010101", -- Y Z \_

"010111010111010111", "01110111010101110111"); -- . ,

if (Print\_Word = '1') then

SigPrint <= '1';

if letters = 'A' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(0));

elsif Letters = 'B' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(1));

elsif Letters = 'C' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(2));

elsif letters = 'D' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(3));

elsif Letters = 'E' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(4));

elsif Letters = 'F' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(5));

elsif Letters = 'G' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(6));

elsif Letters = 'H' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(7));

elsif Letters = 'I' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(8));

elsif Letters = 'J' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(9));

elsif Letters = 'K' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(10));

elsif Letters = 'L' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(11));

elsif Letters = 'M' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(12));

elsif Letters = 'N' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(13));

elsif Letters = 'O' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(14));

elsif Letters = 'P' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(15));

elsif Letters = 'Q' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(16));

elsif Letters = 'R' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(17));

elsif Letters = 'S' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(18));

elsif Letters = 'T' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(19));

elsif Letters = 'U' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(20));

elsif Letters = 'V' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(21));

elsif Letters = 'W' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(22));

elsif Letters = 'X' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(23));

elsif Letters = 'Y' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(24));

elsif Letters = 'Z' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(25));

elsif Letters = '\_' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(26));

elsif Letters = '.' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(27));

elsif Letters = ',' then

MasterList <= STD\_LOGIC\_VECTOR(MorseArray(28));

end if;

end if;

end process Translator;

end Behavioral;

Constraints Code

set\_property PACKAGE\_PIN V17 [get\_ports {SwitchBus[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SwitchBus[0]}]

set\_property PACKAGE\_PIN V16 [get\_ports {SwitchBus[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SwitchBus[1]}]

set\_property PACKAGE\_PIN W16 [get\_ports {SwitchBus[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SwitchBus[2]}]

set\_property PACKAGE\_PIN W17 [get\_ports {SwitchBus[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SwitchBus[3]}]

set\_property PACKAGE\_PIN W15 [get\_ports {SwitchBus[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SwitchBus[4]}]

set\_property PACKAGE\_PIN V15 [get\_ports {SwitchBus[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SwitchBus[5]}]

set\_property PACKAGE\_PIN P1 [get\_ports {SigLED}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SigLED}]

set\_property PACKAGE\_PIN U18 [get\_ports {New\_Letter}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {New\_Letter}]

set\_property PACKAGE\_PIN R2 [get\_ports {Print\_Word}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Print\_Word}]

set\_property PACKAGE\_PIN L1 [get\_ports {SigPrint}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SigPrint}]

set\_property PACKAGE\_PIN U16 [get\_ports {MorseCode[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[0]}]

set\_property PACKAGE\_PIN E19 [get\_ports {MorseCode[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[1]}]

set\_property PACKAGE\_PIN U19 [get\_ports {MorseCode[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[2]}]

set\_property PACKAGE\_PIN V19 [get\_ports {MorseCode[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[3]}]

set\_property PACKAGE\_PIN W18 [get\_ports {MorseCode[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[4]}]

set\_property PACKAGE\_PIN U15 [get\_ports {MorseCode[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[5]}]

set\_property PACKAGE\_PIN U14 [get\_ports {MorseCode[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[6]}]

set\_property PACKAGE\_PIN V14 [get\_ports {MorseCode[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[7]}]

set\_property PACKAGE\_PIN V13 [get\_ports {MorseCode[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[8]}]

set\_property PACKAGE\_PIN V3 [get\_ports {MorseCode[9]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[9]}]

set\_property PACKAGE\_PIN W3 [get\_ports {MorseCode[10]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[10]}]

set\_property PACKAGE\_PIN U3 [get\_ports {MorseCode[11]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[11]}]

set\_property PACKAGE\_PIN P3 [get\_ports {MorseCode[12]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[12]}]

set\_property PACKAGE\_PIN N3 [get\_ports {MorseCode[13]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {MorseCode[13]}]

set\_property PACKAGE\_PIN W5 [get\_ports {Clock}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Clock}]

create\_clock -add -name sys\_clk\_pin - period 10 -waveform [0 5] [get\_ports {Clock}]

#####################################################

set\_property PACKAGE\_PIN W4 [get\_ports {Anode[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Anode[3]}]

set\_property PACKAGE\_PIN V4 [get\_ports {Anode[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Anode[2]}]

set\_property PACKAGE\_PIN U4 [get\_ports {Anode[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Anode[1]}]

set\_property PACKAGE\_PIN U2 [get\_ports {Anode[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Anode[0]}]

######################################################

set\_property PACKAGE\_PIN W7 [get\_ports {Cathode[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Cathode[0]}]

set\_property PACKAGE\_PIN W6 [get\_ports {Cathode[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Cathode[1]}]

set\_property PACKAGE\_PIN U8 [get\_ports {Cathode[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Cathode[2]}]

set\_property PACKAGE\_PIN V8 [get\_ports {Cathode[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Cathode[3]}]

set\_property PACKAGE\_PIN U5 [get\_ports {Cathode[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Cathode[4]}]

set\_property PACKAGE\_PIN V5 [get\_ports {Cathode[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Cathode[5]}]

set\_property PACKAGE\_PIN U7 [get\_ports {Cathode[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Cathode[6]}]

“\_” - 0

A - 1

B - 2

C - 3

D - 4  
E - 5

F - 6

G - 7

H - 8

I - 9

J - 10

K - 11

L - 12  
M - 13  
N - 14

O - 15  
P - 16

Q - 17  
R - 18  
S - 19  
T - 20  
U - 21  
V - 22  
W - 23  
X - 24  
Y - 25  
Z - 26

“.” - 27

“,” - 28

“\_” …….

A .-

B -...

C -.-.

D -..  
E .

F ..-.

G --.

H ….

I ..

J .---

K -.-

L .-..  
M --  
N -.

O ---  
P .--.

Q --.-  
R .-.  
S ...  
T -  
U ..-  
V ...-  
W .--  
X -..-  
Y -.--  
Z --..

“.” .-.-.-

“,” --..--