3to8 Decoder:

Behavioural:

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 primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity decoder\_3to8 is

Port ( A0 : in STD\_LOGIC;

A1 : in STD\_LOGIC;

A2 : in STD\_LOGIC;

Q0 : out STD\_LOGIC;

Q1 : out STD\_LOGIC;

Q2 : out STD\_LOGIC;

Q3 : out STD\_LOGIC;

Q4 : out STD\_LOGIC;

Q5 : out STD\_LOGIC;

Q6 : out STD\_LOGIC;

Q7 : out STD\_LOGIC);

end decoder\_3to8;

architecture Behavioral of decoder\_3to8 is

begin

Q0<= ((not A0) and (not A1) and (not A2)) after 5 ns;

Q1<= (( not A0) and (not A1) and A2) after 5 ns;

Q2<= ((not A0) and A1 and (not A2)) after 5 ns;

Q3<= ((not A0) and A1 and A2) after 5 ns;

Q4<= (A0 and (not A1) and (not A2)) after 5 ns;

Q5<= (A0 and (not A1) and A2) after 5 ns;

Q6<= (A0 and A1 and (not A2)) after 5 ns;

Q7<= (A0 and A1 and A2) after 5 ns;

end Behavioral;

Test Bench:

COMPONENT decoder\_3to8

PORT(

A0 : IN std\_logic;

A1 : IN std\_logic;

A2 : IN std\_logic;

Q0 : OUT std\_logic;

Q1 : OUT std\_logic;

Q2 : OUT std\_logic;

Q3 : OUT std\_logic;

Q4 : OUT std\_logic;

Q5 : OUT std\_logic;

Q6 : OUT std\_logic;

Q7 : OUT std\_logic

);

END COMPONENT;

--Inputs

signal A0 : std\_logic := '0';

signal A1 : std\_logic := '0';

signal A2 : std\_logic := '0';

--Outputs

signal Q0 : std\_logic;

signal Q1 : std\_logic;

signal Q2 : std\_logic;

signal Q3 : std\_logic;

signal Q4 : std\_logic;

signal Q5 : std\_logic;

signal Q6 : std\_logic;

signal Q7 : std\_logic;

BEGIN

uut: decoder\_3to8 PORT MAP (

A0 => A0,

A1 => A1,

A2 => A2,

Q0 => Q0,

Q1 => Q1,

Q2 => Q2,

Q3 => Q3,

Q4 => Q4,

Q5 => Q5,

Q6 => Q6,

Q7 => Q7

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 100 ns;

A0 <= (not A0);

wait for 50 ns;

A1 <= (not A1);

wait for 25 ns;

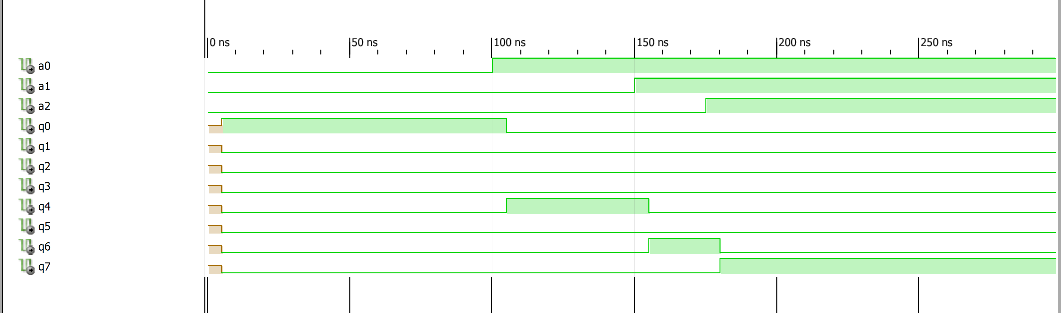
A2 <= (not A2);

wait;

end process;

END;

Screen Shot:



Multiplexer 2to16:

Behavioural:

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 primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity mux2to16 is

Port ( s : in STD\_LOGIC;

In0 : in STD\_LOGIC\_VECTOR (15 downto 0);

In1 : in STD\_LOGIC\_VECTOR (15 downto 0);

Z : out STD\_LOGIC\_VECTOR (15 downto 0));

end mux2to16;

architecture Behavioral of mux2to16 is

begin

Z <= In0 after 5 ns when S='0' else

In1 after 5 ns when S='1'else

"0000000000000000" after 5 ns;

end Behavioral;

Test Bench:

COMPONENT mux2to16

PORT(

s : IN std\_logic;

In0 : IN std\_logic\_vector(15 downto 0);

In1 : IN std\_logic\_vector(15 downto 0);

Z : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal s : std\_logic := '0';

signal In0 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In1 : std\_logic\_vector(15 downto 0) := (others => '0');

--Outputs

signal Z : std\_logic\_vector(15 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: mux2to16 PORT MAP (

s => s,

In0 => In0,

In1 => In1,

Z => Z

);

-- Stimulus process

stim\_proc: process

begin

In0 <= "0000000000000000";

In1 <= "1000000000000000";

wait for 15 ns;

s <= (not s);

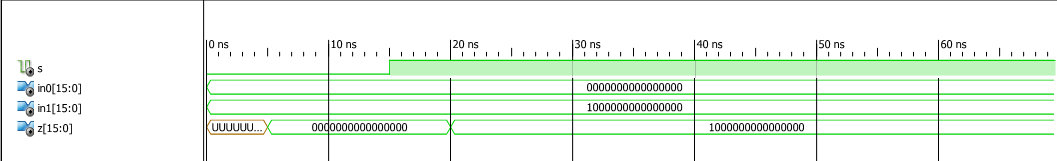
wait;

wait;

end process;

END;

Screen Shot:



MUX 8to16 Bit:

Behavioural:

entity mux\_2to8 is

Port ( S0 : in STD\_LOGIC;

S1 : in STD\_LOGIC;

S2 : in STD\_LOGIC;

In0 : in STD\_LOGIC\_VECTOR (15 downto 0);

In1 : in STD\_LOGIC\_VECTOR (15 downto 0);

In2 : in STD\_LOGIC\_VECTOR (15 downto 0);

In3 : in STD\_LOGIC\_VECTOR (15 downto 0);

In4 : in STD\_LOGIC\_VECTOR (15 downto 0);

In5 : in STD\_LOGIC\_VECTOR (15 downto 0);

In6 : in STD\_LOGIC\_VECTOR (15 downto 0);

In7 : in STD\_LOGIC\_VECTOR (15 downto 0);

Z : out STD\_LOGIC\_VECTOR (15 downto 0));

end mux\_2to8;

architecture Behavioral of mux\_2to8 is

begin

Z <= In0 after 5 ns when S0='0' and S1='0' and S1='1' else

In1 after 5 ns when S0='1' and S1='0' and S2='0' else

In2 after 5 ns when S0='0' and S1='1' and S2 ='0' else

In3 after 5 ns when S0='1' and S1='1' and S2 ='0' else

In4 after 5 ns when S0='0' and S1='0' and S2 ='1' else

In5 after 5 ns when S0='1' and S1='0' and S2 ='1' else

In6 after 5 ns when S0='0' and S1='1' and S2 ='1' else

In7 after 5 ns when S0='1' and S1='1' and S2 ='1' else

"0000000000000000" after 5 ns;

end Behavioral;

Test Bench:

COMPONENT mux\_2to8

PORT(

S0 : IN std\_logic;

S1 : IN std\_logic;

S2 : IN std\_logic;

In0 : IN std\_logic\_vector(15 downto 0);

In1 : IN std\_logic\_vector(15 downto 0);

In2 : IN std\_logic\_vector(15 downto 0);

In3 : IN std\_logic\_vector(15 downto 0);

In4 : IN std\_logic\_vector(15 downto 0);

In5 : IN std\_logic\_vector(15 downto 0);

In6 : IN std\_logic\_vector(15 downto 0);

In7 : IN std\_logic\_vector(15 downto 0);

Z : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal S0 : std\_logic := '0';

signal S1 : std\_logic := '0';

signal S2 : std\_logic := '0';

signal In0 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In1 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In2 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In3 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In4 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In5 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In6 : std\_logic\_vector(15 downto 0) := (others => '0');

signal In7 : std\_logic\_vector(15 downto 0) := (others => '0');

--Outputs

signal Z : std\_logic\_vector(15 downto 0);

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: mux\_2to8 PORT MAP (

S0 => S0,

S1 => S1,

S2 => S2,

In0 => In0,

In1 => In1,

In2 => In2,

In3 => In3,

In4 => In4,

In5 => In5,

In6 => In6,

In7 => In7,

Z => Z

);

-- Stimulus process

stim\_proc: process

begin

In0 <= "0000000000000000";

In1 <= "1000000000000000";

In2 <= "1100000000000000";

In3 <= "1110000000000000";

In4 <= "1111000000000000";

In5 <= "1111100000000000";

In6 <= "1111110000000000";

In7 <= "1111111000000000";

wait for 15 ns;

S0 <= (not S0);

wait for 10 ns;

S1 <= (not S1);

wait for 5 ns;

S2 <= (not S2);

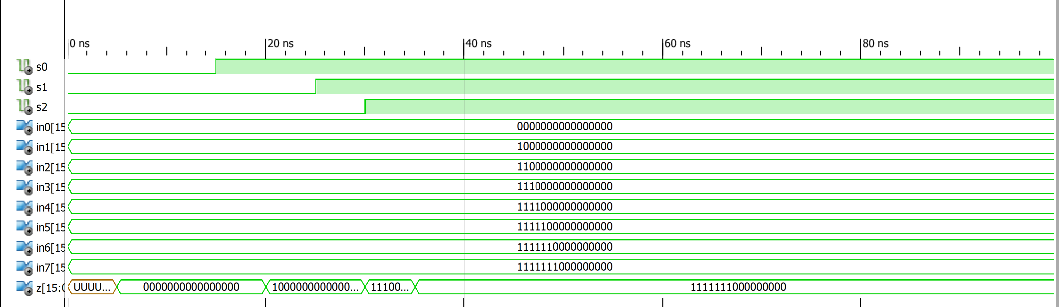
wait;

wait;

end process;

END;

Screen Shot:



Register:

Behavioural:

entity reg16 is

Port ( D : in STD\_LOGIC\_VECTOR (15 downto 0);

load : in STD\_LOGIC;

CLK : in STD\_LOGIC;

Q : out STD\_LOGIC\_VECTOR (15 downto 0));

end reg16;

architecture Behavioral of reg16 is

begin

process(CLK)

begin

if (rising\_edge(CLK)) then

if load='1' then

Q<=D after 5 ns; end if;

end if;

end process;

end Behavioral;

Test Bench:

COMPONENT reg16

PORT(

D : IN std\_logic\_vector(15 downto 0);

load : IN std\_logic;

CLK : IN std\_logic;

Q : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal D : std\_logic\_vector(15 downto 0) := (others => '0');

signal load : std\_logic := '0';

signal CLK : std\_logic := '0';

--Outputs

signal Q : std\_logic\_vector(15 downto 0);

-- Clock period definitions

constant CLK\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: reg16 PORT MAP (

D => D,

load => load,

CLK => CLK,

Q => Q

);

-- Clock process definitions

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

wait for CLK\_period\*10;

load <= '1';

wait for 5 ns;

D <= "0000001010000000";

wait for CLK\_period\*10;

load <= '0';

wait for 5 ns;

D <= "1111111111000000";

wait for CLK\_period\*10;

load <='1';

wait for 5 ns;

D <= "1010101010101010";

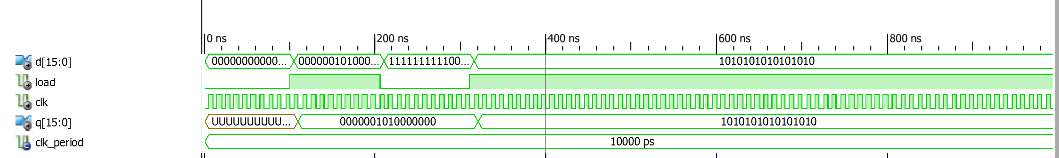
-- insert stimulus here

wait;

end process;

END;

Screenshot:



Register File:

entity reg\_file is

Port ( src\_s0 : in STD\_LOGIC;

src\_s1 : in STD\_LOGIC;

src\_s2 : in STD\_logic;

des\_A0 : in STD\_LOGIC;

des\_A1 : in STD\_LOGIC;

des\_A2 : in STD\_lOGIC;

CLK : in STD\_LOGIC;

data\_src : in STD\_LOGIC;

data : in STD\_LOGIC\_VECTOR (15 downto 0);

reg0 : out STD\_LOGIC\_VECTOR (15 downto 0);

reg1 : out STD\_LOGIC\_VECTOR (15 downto 0);

reg2 : out STD\_LOGIC\_VECTOR (15 downto 0);

reg3 : out STD\_LOGIC\_VECTOR (15 downto 0);

reg4 : out STD\_LOGIC\_VECTOR (15 downto 0);

reg5 : out STD\_LOGIC\_VECTOR (15 downto 0);

reg6 : out STD\_LOGIC\_VECTOR (15 downto 0);

reg7 : out STD\_LOGIC\_VECTOR (15 downto 0));

end reg\_file;

architecture Behavioral of reg\_file is

-- 4 bit Register for register file

COMPONENT reg16

PORT(

D : IN std\_logic\_vector(15 downto 0);

load : IN std\_logic;

CLK : IN std\_logic;

Q : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

-- 2 to 4 Decoder

COMPONENT decoder\_3to8

PORT(

A0 : IN std\_logic;

A1 : IN std\_logic;

A2 : IN std\_logic;

Q0 : OUT std\_logic;

Q1 : OUT std\_logic;

Q2 : OUT std\_logic;

Q3 : OUT std\_logic;

Q4 : OUT std\_logic;

Q5 : OUT std\_logic;

Q6 : OUT std\_logic;

Q7 : OUT std\_logic

);

END COMPONENT;

-- 2 to 1 line multiplexer

COMPONENT mux2to16

PORT(

s : IN std\_logic;

In0 : IN std\_logic\_vector(15 downto 0);

In1 : IN std\_logic\_vector(15 downto 0);

Z : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

-- 4 to 1 line multiplexer

COMPONENT mux\_2to8

PORT( S0 : IN std\_logic;

S1 : IN std\_logic;

S2 : IN std\_logic;

In0 : IN std\_logic\_vector(15 downto 0);

In1 : IN std\_logic\_vector(15 downto 0);

In2 : IN std\_logic\_vector(15 downto 0);

In3 : IN std\_logic\_vector(15 downto 0);

In4 : IN std\_logic\_vector(15 downto 0);

In5 : IN std\_logic\_vector(15 downto 0);

In6 : IN std\_logic\_vector(15 downto 0);

In7 : IN std\_logic\_vector(15 downto 0);

Z : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

-- signals

signal load\_reg0, load\_reg1, load\_reg2, load\_reg3, load\_reg4, load\_reg5, load\_reg6, load\_reg7 : std\_logic;

signal reg0\_q, reg1\_q, reg2\_q, reg3\_q, reg4\_q, reg5\_q, reg6\_q, reg7\_q,

data\_src\_mux\_out, src\_reg : std\_logic\_vector(15 downto 0);

begin

-- port maps ;-)

-- register 0

reg00: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg0,

CLK => CLK,

Q => reg0\_q

);

-- register 1

reg01: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg1,

CLK => CLK,

Q => reg1\_q

);

-- register 2

reg02: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg2,

CLK => CLK,

Q => reg2\_q

);

-- register 3

reg03: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg3,

CLK => CLK,

Q => reg3\_q

);

-- register 4

reg04: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg4,

CLK => CLK,

Q => reg4\_q

);

-- register 5

reg05: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg5,

CLK => CLK,

Q => reg5\_q

);

-- register 6

reg06: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg6,

CLK => CLK,

Q => reg6\_q

);

-- register 7

reg07: reg16 PORT MAP(

D => data\_src\_mux\_out,

load => load\_reg7,

CLK => CLK,

Q => reg7\_q

);

-- Destination register decoder

des\_decoder :decoder\_3to8 PORT MAP(

A0 => des\_A0,

A1 => des\_A1,

A2 => des\_A2,

Q0 => load\_reg0,

Q1 => load\_reg1,

Q2 => load\_reg2,

Q3 => load\_reg3,

Q4 => load\_reg4,

Q5 => load\_reg5,

Q6 => load\_reg6,

Q7 => load\_reg7

);

-- 2 to 1 Data source multiplexer

data\_2to16MUX: mux2to16 PORT MAP(

In0 => data,

In1 => src\_reg,

s => data\_src,

Z => data\_src\_mux\_out

);

-- 4 to 1 source register multiplexer

dstMUX2to16: mux\_2to8 PORT MAP(

In0 => reg0\_q,

In1 => reg1\_q,

In2 => reg2\_q,

In3 => reg3\_q,

In4 => reg4\_q,

In5 => reg5\_q,

In6 => reg6\_q,

In7 => reg7\_q,

S0 => src\_s0,

S1 => src\_s1,

S2 => src\_s2,

Z => src\_reg

);

reg0 <= reg0\_q;

reg1 <= reg1\_q;

reg2 <= reg2\_q;

reg3 <= reg3\_q;reg4 <= reg4\_q;

reg5 <= reg5\_q;

reg6 <= reg6\_q;

reg7 <= reg7\_q;

end Behavioral;

Test Bench:

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;

ENTITY regFile\_tb IS

END regFile\_tb;

ARCHITECTURE behavior OF regFile\_tb IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT reg\_file

PORT(

src\_s0 : IN std\_logic;

src\_s1 : IN std\_logic;

src\_s2 : IN std\_logic;

des\_A0 : IN std\_logic;

des\_A1 : IN std\_logic;

des\_A2 : IN std\_logic;

CLK : IN std\_logic;

data\_src : IN std\_logic;

data : IN std\_logic\_vector(15 downto 0);

reg0 : OUT std\_logic\_vector(15 downto 0);

reg1 : OUT std\_logic\_vector(15 downto 0);

reg2 : OUT std\_logic\_vector(15 downto 0);

reg3 : OUT std\_logic\_vector(15 downto 0);

reg4 : OUT std\_logic\_vector(15 downto 0);

reg5 : OUT std\_logic\_vector(15 downto 0);

reg6 : OUT std\_logic\_vector(15 downto 0);

reg7 : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal src\_s0 : std\_logic := '0';

signal src\_s1 : std\_logic := '0';

signal src\_s2 : std\_logic := '0';

signal src\_s3 : std\_logic := '0';

signal des\_A0 : std\_logic := '0';

signal des\_A1 : std\_logic := '0';

signal des\_A2 : std\_logic := '0';

signal CLK : std\_logic := '0';

signal data\_src : std\_logic := '0';

signal data : std\_logic\_vector(15 downto 0) := (others => '0');

--Outputs

signal reg0 : std\_logic\_vector(15 downto 0);

signal reg1 : std\_logic\_vector(15 downto 0);

signal reg2 : std\_logic\_vector(15 downto 0);

signal reg3 : std\_logic\_vector(15 downto 0);

signal reg4 : std\_logic\_vector(15 downto 0);

signal reg5 : std\_logic\_vector(15 downto 0);

signal reg6 : std\_logic\_vector(15 downto 0);

signal reg7 : std\_logic\_vector(15 downto 0);

-- Clock period definitions

constant CLK\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: reg\_file PORT MAP (

src\_s0 => src\_s0,

src\_s1 => src\_s1,

src\_s2 => src\_s2,

des\_A0 => des\_A0,

des\_A1 => des\_A1,

des\_A2 => des\_A2,

CLK => CLK,

data\_src => data\_src,

data => data,

reg0 => reg0,

reg1 => reg1,

reg2 => reg2,

reg3 => reg3,

reg4 => reg4,

reg5 => reg5,

reg6 => reg6,

reg7 => reg7

);

-- Clock process definitions

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

wait for 100 ns;

data\_src <= '0';

data <= "0000000000000000";

des\_A0 <= '0';

des\_A1 <= '0';

des\_A2 <= '0';

wait for CLK\_period\*10;

data <= "0000000000000001";

des\_A0 <= '0';

des\_A1 <= '0';

des\_A2 <= '1';

wait for CLK\_period\*10;

data <= "0000000000000010";

des\_A0 <= '0';

des\_A1 <= '1';

des\_A2 <= '0';

wait for CLK\_period\*10;

data <= "0000000000000011";

des\_A0 <= '0';

des\_A1 <= '1';

des\_A2 <= '1';

wait for CLK\_period\*10;

data <= "0000000000000100";

des\_A0 <= '1';

des\_A1 <= '0';

des\_A2 <= '0';

wait for CLK\_period\*10;

data <= "0000000000000101";

des\_A0 <= '1';

des\_A1 <= '0';

des\_A2 <= '1';

wait for CLK\_period\*10;

data <= "0000000000000111";

des\_A0 <= '1';

des\_A1 <= '1';

des\_A2 <= '0';

wait for CLK\_period\*10;

data <= "0000000000001000";

des\_A0 <= '1';

des\_A1 <= '1';

des\_A2 <= '1';

wait for CLK\_period\*10;

data\_src <= '1';

des\_A0 <= '1';

des\_A1 <='1';

des\_A2 <='1';

src\_s0 <= '0';

src\_s1 <= '0';

src\_s2 <= '0';

end process;

END;

ScreenShot:

