**CS2022 PROJECT 1- DATAPATH DESIGN**

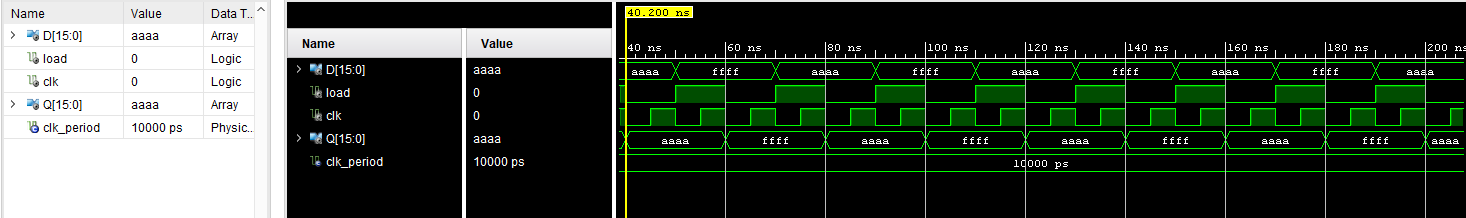
**PART A**

Aniket Agarwal 17317437

**RESULTS OF TEST BENCHES**

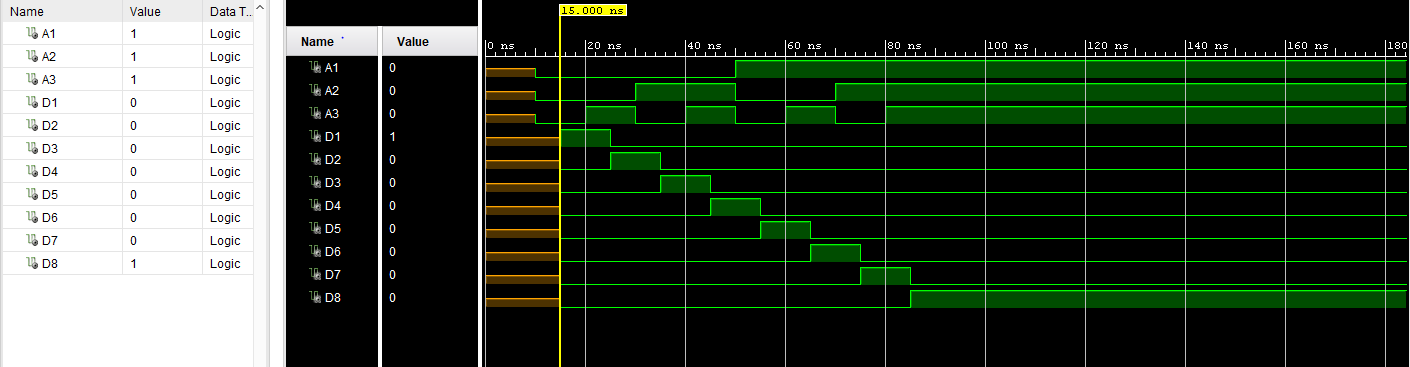
**Register**

The register is given a new value after every 5ns. It works as expected for each edge on a given rising clock signal where the load is high.



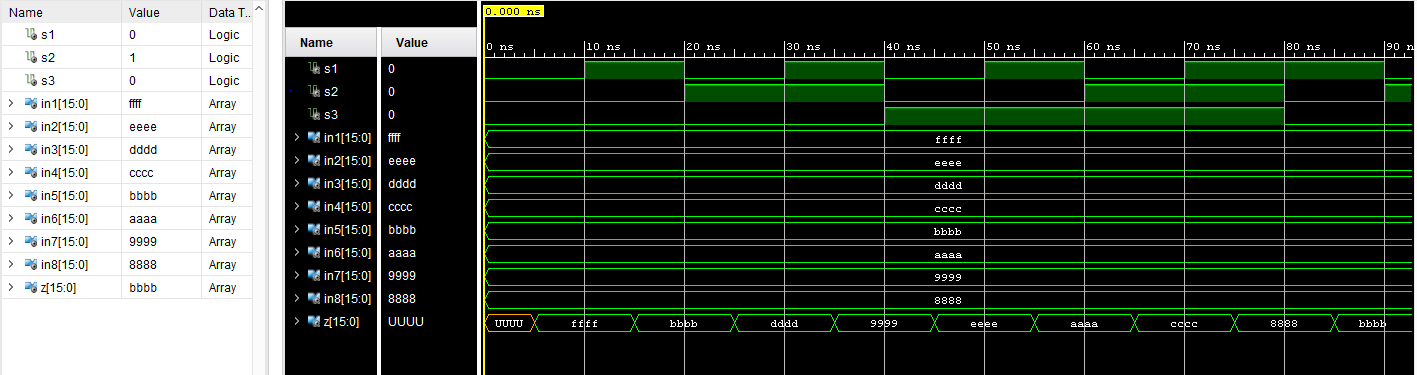
**Decoder 3\_to\_8 16 bit**

The test bench cycles through each given combination, causing each given output pin to turn high via the register selection.



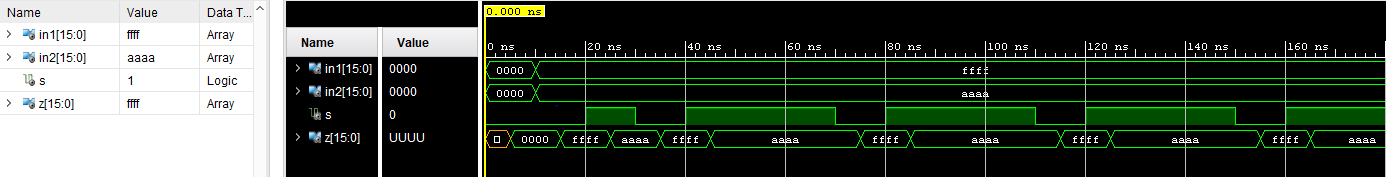
**Multiplexer 8\_to\_1 16 bit**

This multiplexer cycles through all given combinations of s1, s2 and s3, thus allowing z to cycle through the inputs in1 to in8 as the output pin.



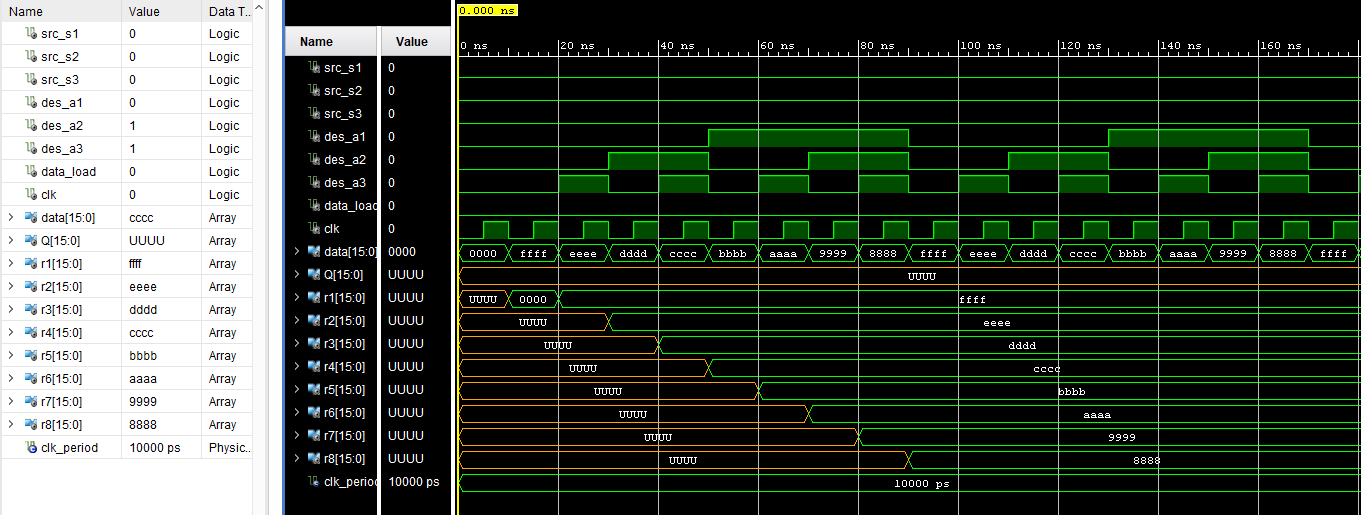
**Multiplexer 2\_to\_1 16 bit**

The multiplexer cycles through a series of changes via s between 0 and 1. The output of the multiplexer switches between the given outputs for in1 and in2.

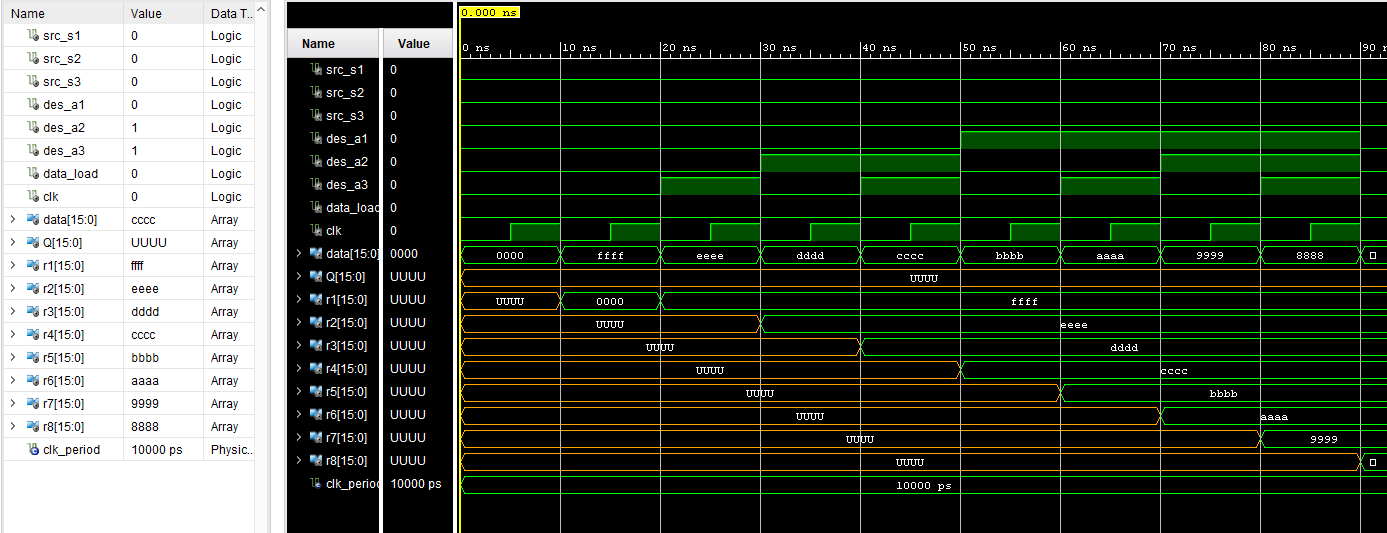


**Final VHDL Code**

Register simulation given different hexadecimal value for each subsequent data input, i.e. Ri<-Xi i = 1; 8



The register r1 is loaded with values from the input data from the test bench. The value is then transferred from one given register to the incremented register, r2; i.e. from r1 to r2, r2 to r3 ... r7 to r8. As each load is cycled through the data inputs, the transfer of data cycling between each register is passed on until the input is reset to 0x0000 after the first input time, which shows that the values stem from the registers, and not from the data input itself sequentially, i.e. Ri<-Rj i; j = 0; 7



**VHDL Components Source Codes**

**Register**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity reg is

Port(

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

load, clk : in STD\_LOGIC;

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

);

end reg;

architecture Behavioral of reg is

begin process(clk)

begin

if(rising\_edge(clk)) then

if(load = '1') then

Q <= D after 5ns;

end if;

end if;

end process;

end Behavioral;

**Decoder 3\_to\_8 16 bit**

library IEEE;

use ieee.std\_logic\_1164.all;

entity decoder is

port(A1, A2, A3: in std\_logic;

D1, D2, D3, D4, D5, D6, D7, D8: out std\_logic);

end decoder;

architecture Behavioral of decoder is

begin

D1<=((not A1) and (not A2) and (not A3)) after 5ns ;

D2<=((not A1) and (not A2) and (A3)) after 5ns ;

D3<=((not A1) and (A2) and (not A3)) after 5ns ;

D4<=((not A1) and (A2) and (A3)) after 5ns ;

D5<=((A1) and (not A2) and (not A3)) after 5ns ;

D6<=((A1) and (not A2) and (A3)) after 5ns ;

D7<=((A1) and (A2) and (not A3)) after 5ns ;

D8<=((A1) and (A2) and (A3)) after 5ns ;

end Behavioral;

**Multiplexer 8\_to\_1 16 bit**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity multiplexer is

Port ( s1, s2, s3 : in STD\_LOGIC;

in1, in2, in3, in4, in5, in6, in7, in8 : in STD\_LOGIC\_VECTOR (15 downto 0);

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

end multiplexer;

architecture Behavioral of multiplexer is

begin

z <= in1 after 5ns when s1 = '0' and s2 = '0' and s3 = '0' else

in2 after 5ns when s1 = '0' and s2 = '0' and s3 = '1' else

in3 after 5ns when s1 = '0' and s2 = '1' and s3 = '0' else

in4 after 5ns when s1 = '0' and s2 = '1' and s3 = '1' else

in5 after 5ns when s1 = '1' and s2 = '0' and s3 = '0' else

in6 after 5ns when s1 = '1' and s2 = '0' and s3 = '1' else

in7 after 5ns when s1 = '1' and s2 = '1' and s3 = '0' else

in8 after 5ns when s1 = '1' and s2 = '1' and s3 = '1' else

x"0000" after 5ns;

end Behavioral;

**Multiplexer 2\_to\_1 16 bit**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux is

Port(

s : in STD\_LOGIC;

in1, in2 : in STD\_LOGIC\_VECTOR(15 downto 0);

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end mux;

architecture Behavioral of mux is

begin

z <= in1 after 5ns when s = '0' else

in2 after 5ns when s = '1' else

x"0000" after 5ns;

end Behavioral;

**Final VHDL code**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity register\_final is

Port(

src\_s1, src\_s2, src\_s3 : in STD\_LOGIC;

des\_a1, des\_a2, des\_a3 : in STD\_LOGIC;

data\_load, clk : in STD\_LOGIC;

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

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

r1, r2, r3, r4, r5, r6, r7, r8 : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end register\_final;

architecture Behavioral of register\_final is

Component reg

Port(

load, clk : in STD\_LOGIC;

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

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

);

End Component;

Component decoder

Port(

a1, a2, a3 : in STD\_LOGIC;

D1, D2, D3, D4, D5, D6, D7, D8 : out STD\_LOGIC

);

End Component;

Component mux

Port(

in1, in2 : in STD\_LOGIC\_VECTOR(15 downto 0);

s : in STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component multiplexer

Port(

in1, in2, in3, in4, in5, in6, in7, in8 : in STD\_LOGIC\_VECTOR(15 downto 0);

s1, s2, s3 : STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

signal load\_r1, load\_r2, load\_r3, load\_r4, load\_r5, load\_r6, load\_r7, load\_r8 : STD\_LOGIC;

signal q\_r1, q\_r2, q\_r3, q\_r4, q\_r5, q\_r6, q\_r7, q\_r8 : STD\_LOGIC\_VECTOR(15 downto 0);

signal data\_src\_mux\_out, src\_r : STD\_LOGIC\_VECTOR(15 downto 0);

begin

reg1 : reg PORT MAP(

load => load\_r1,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r1);

reg2 : reg PORT MAP(

load => load\_r2,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r2

);

reg3 : reg PORT MAP(

load => load\_r3,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r3

);

reg4 : reg PORT MAP(

load => load\_r4,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r4

);

reg5 : reg PORT MAP(

load => load\_r5,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r5

);

reg6 : reg PORT MAP(

load => load\_r6,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r6

);

reg7 : reg PORT MAP(

load => load\_r7,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r7 );

reg8 : reg PORT MAP(

load => load\_r8,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r8

);

decoder\_3\_8 : decoder PORT MAP(

a1 => des\_a1,

a2 => des\_a2,

a3 => des\_a3,

D1 => load\_r1,

D2 => load\_r2,

D3 => load\_r3,

D4 => load\_r4,

D5 => load\_r5,

D6 => load\_r6,

D7 => load\_r7,

D8 => load\_r8

);

mux\_2\_16 : mux PORT MAP(

in1 => data,

in2 => src\_r,

s => data\_load,

z => data\_src\_mux\_out

);

mux\_8\_16 : multiplexer PORT MAP(

in1 => q\_r1,

in2 => q\_r2,

in3 => q\_r3,

in4 => q\_r4,

in5 => q\_r5,

in6 => q\_r6,

in7 => q\_r7,

in8 => q\_r8,

s1 => src\_s1,

s2 => src\_s2,

s3 => src\_s3,

z => src\_r

);

r1 <= q\_r1;

r2 <= q\_r2;

r3 <= q\_r3;

r4 <= q\_r4;

r5 <= q\_r5;

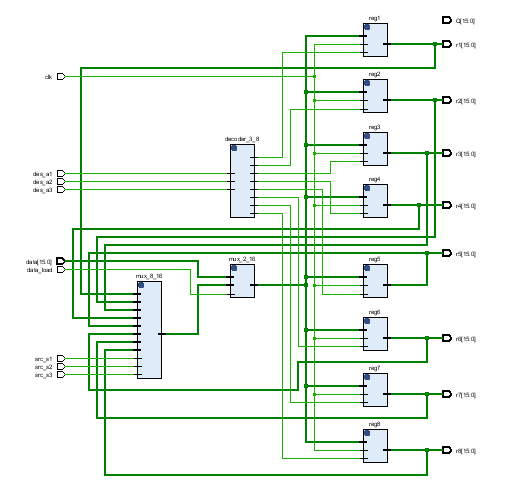
r6 <= q\_r6;

r7 <= q\_r7;

r8 <= q\_r8;

end Behavioral;

**Schematics for the circuit**



**Component Test Benches**

**Register**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity register\_final is

Port(

src\_s1, src\_s2, src\_s3 : in STD\_LOGIC;

des\_a1, des\_a2, des\_a3 : in STD\_LOGIC;

data\_load, clk : in STD\_LOGIC;

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

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

r1, r2, r3, r4, r5, r6, r7, r8 : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end register\_final;

architecture Behavioral of register\_final is

Component reg

Port(

load, clk : in STD\_LOGIC;

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

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

);

End Component;

Component decoder

Port(

a1, a2, a3 : in STD\_LOGIC;

D1, D2, D3, D4, D5, D6, D7, D8 : out STD\_LOGIC

);

End Component;

Component mux

Port(

in1, in2 : in STD\_LOGIC\_VECTOR(15 downto 0);

s : in STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component multiplexer

Port(

in1, in2, in3, in4, in5, in6, in7, in8 : in STD\_LOGIC\_VECTOR(15 downto 0);

s1, s2, s3 : STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

signal load\_r1, load\_r2, load\_r3, load\_r4, load\_r5, load\_r6, load\_r7, load\_r8 : STD\_LOGIC;

signal q\_r1, q\_r2, q\_r3, q\_r4, q\_r5, q\_r6, q\_r7, q\_r8 : STD\_LOGIC\_VECTOR(15 downto 0);

signal data\_src\_mux\_out, src\_r : STD\_LOGIC\_VECTOR(15 downto 0);

begin

reg1 : reg PORT MAP(

load => load\_r1,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r1

);

reg2 : reg PORT MAP(

load => load\_r2,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r2 );

reg3 : reg PORT MAP(

load => load\_r3,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r3 );

reg4 : reg PORT MAP(

load => load\_r4,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r4 );

reg5 : reg PORT MAP(

load => load\_r5,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r5 );

reg6 : reg PORT MAP(

load => load\_r6,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r6 );

reg7 : reg PORT MAP(

load => load\_r7,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r7 );

reg8 : reg PORT MAP(

load => load\_r8,

clk => clk,

D => data\_src\_mux\_out,

Q => q\_r8 );

decoder\_3\_8 : decoder PORT MAP(

a1 => des\_a1,

a2 => des\_a2,

a3 => des\_a3,

D1 => load\_r1,

D2 => load\_r2,

D3 => load\_r3,

D4 => load\_r4,

D5 => load\_r5,

D6 => load\_r6,

D7 => load\_r7,

D8 => load\_r8 );

mux\_2\_16 : mux PORT MAP(

in1 => data,

in2 => src\_r,

s => data\_load,

z => data\_src\_mux\_out);

mux\_8\_16 : multiplexer PORT MAP(

in1 => q\_r1,

in2 => q\_r2,

in3 => q\_r3,

in4 => q\_r4,

in5 => q\_r5,

in6 => q\_r6,

in7 => q\_r7,

in8 => q\_r8,

s1 => src\_s1,

s2 => src\_s2,

s3 => src\_s3,

z => src\_r );

r1 <= q\_r1;r2 <= q\_r2; r3 <= q\_r3;r4 <= q\_r4; r5 <= q\_r5;r6 <= q\_r6; r7 <= q\_r7;r8 <= q\_r8;

end Behavioral;

**Decoder 3\_to\_8 16 bit**

library IEEE;

use IEEE.Std\_logic\_1164.all;

use IEEE.Numeric\_Std.all;

entity decoder\_tb is

end;

architecture behavior of decoder\_tb is

component decoder

port(A1, A2, A3: in std\_logic;

D1, D2, D3, D4, D5, D6, D7, D8: out std\_logic);

end component;

signal A1, A2, A3: std\_logic;

signal D1, D2, D3, D4, D5, D6, D7, D8: std\_logic;

begin

uut: decoder port map ( A1 => A1,A2 => A2, A3 => A3,

D1 => D1,D2 => D2,D3 => D3,D4 => D4,D5 => D5,D6 => D6,D7 => D7,D8 => D8

);

stimulus: process

begin

wait for 10ns; A1 <= '0';A2 <= '0';A3 <= '0';

wait for 10ns; A1 <= '0';A2 <= '0';A3 <= '1';

wait for 10ns; A1 <= '0';A2 <= '1';A3 <= '0';

wait for 10ns; A1 <= '0';A2 <= '1';A3 <= '1';

wait for 10ns; A1 <= '1';A2 <= '0';A3 <= '0';

wait for 10ns; A1 <= '1';A2 <= '0';A3 <= '1';

wait for 10ns; A1 <= '1';A2 <= '1';A3 <= '0';

wait for 10ns; A1 <= '1';A2 <= '1';A3 <= '1';

wait;

end process;

end;

**Multiplexer 8\_to\_1 16 bit**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY multiplexer\_tb IS

END multiplexer\_tb;

ARCHITECTURE behavior OF multiplexer\_tb IS

COMPONENT multiplexer

PORT(

s1: IN std\_logic;

s2: IN std\_logic;

s3: IN std\_logic;

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);

in8 : IN std\_logic\_vector(15 downto 0);

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

);

END COMPONENT;

signal s1 : std\_logic := '0';

signal s2 : std\_logic := '0';

signal s3 : std\_logic := '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');

signal in8 : std\_logic\_vector(15 downto 0) := (others => '0');

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

BEGIN

uut: multiplexer PORT MAP (

s1 => s1,s2 => s2,s3 => s3,

in1 => in1,in2 => in2,in3 => in3,in4 => in4,in5 => in5,in6 => in6,in7 => in7,in8 => in8,

z => z

);

stim\_proc: process

begin

in1 <= x"FFFF";

in2 <= x"EEEE";

in3 <= x"DDDD";

in4 <= x"CCCC";

in5 <= x"BBBB";

in6 <= x"AAAA";

in7 <= x"9999";

in8 <= x"8888";

wait for 10ns; s1 <= '1';s2 <= '0';s3 <= '0';

wait for 10ns; s1 <= '0';s2 <= '1';s3 <= '0';

wait for 10ns; s1 <= '1';s2 <= '1';s3 <= '0';

wait for 10ns; s1 <= '0';s2 <= '0';s3 <= '1';

wait for 10ns; s1 <= '1';s2 <= '0';s3 <= '1';

wait for 10ns; s1 <= '0';s2 <= '1';s3 <= '1';

wait for 10ns; s1 <= '1';s2 <= '1';s3 <= '1';

end process;

END;

**Multiplexer 2\_to\_1 16 bit**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY mux\_tb IS

END mux\_tb;

ARCHITECTURE behavior OF mux\_tb IS

COMPONENT mux

PORT(

in1 : IN std\_logic\_vector(15 downto 0);

in2 : IN std\_logic\_vector(15 downto 0);

s : IN std\_logic;

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

);

END COMPONENT;

signal in1 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in2 : std\_logic\_vector(15 downto 0) := (others => '0');

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

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

BEGIN

mux\_2\_16: mux PORT MAP (

in1 => in1, in2 => in2,

s => s, z => z );

stim\_proc: process

begin

wait for 10ns; in1 <= x"FFFF"; in2 <= x"AAAA";

wait for 10ns;s <= '1';

wait for 10ns;s <= '0';

wait for 10ns;s <= '1';

end process;

END;

**Final VHDL code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY datapath\_tb IS

END datapath\_tb;

ARCHITECTURE behavior OF datapath\_tb IS

COMPONENT register\_final

PORT(

src\_s1 : IN std\_logic; src\_s2 : IN std\_logic; src\_s3 : IN std\_logic;

des\_a1 : IN std\_logic; des\_a2 : IN std\_logic;des\_a3 : IN std\_logic;

data\_load : IN std\_logic; clk : IN std\_logic;

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

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

r1 : OUT std\_logic\_vector(15 downto 0);

r2 : OUT std\_logic\_vector(15 downto 0);

r3 : OUT std\_logic\_vector(15 downto 0);

r4 : OUT std\_logic\_vector(15 downto 0);

r5 : OUT std\_logic\_vector(15 downto 0);

r6 : OUT std\_logic\_vector(15 downto 0);

r7 : OUT std\_logic\_vector(15 downto 0);

r8 : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

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

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

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

signal des\_a1 : std\_logic := '0';

signal des\_a2 : std\_logic := '0';

signal des\_a3 : std\_logic := '0';

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

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

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

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

signal r1 : std\_logic\_vector(15 downto 0);

signal r2 : std\_logic\_vector(15 downto 0);

signal r3 : std\_logic\_vector(15 downto 0);

signal r4 : std\_logic\_vector(15 downto 0);

signal r5 : std\_logic\_vector(15 downto 0);

signal r6 : std\_logic\_vector(15 downto 0);

signal r7 : std\_logic\_vector(15 downto 0);

signal r8 : std\_logic\_vector(15 downto 0);

BEGIN

uut: register\_final PORT MAP (

src\_s1 => src\_s1, src\_s2 => src\_s2, src\_s3 => src\_s3,

des\_a1 => des\_a1, des\_a2 => des\_a2, des\_a3 => des\_a3,

data\_load => data\_load, clk => clk, data => data,

Q => Q, r1 => r1, r2 => r2, r3 => r3, r4 => r4, r5 => r5, r6 => r6, r7 => r7, r8 => r8);

clk\_process :process

begin

clk <= '0';wait for clk\_period/2; clk <= '1';wait for clk\_period/2;

end process;

stim\_proc: process

begin

wait for 10ns; des\_a1 <= '0'; des\_a2 <= '0'; des\_a3 <= '0'; data <= x"FFFF";

wait for 10ns; des\_a1 <= '0'; des\_a2 <= '0'; des\_a3 <= '1'; data <= x"EEEE";

wait for 10ns; des\_a1 <= '0'; des\_a2 <= '1'; des\_a3 <= '0'; data <= x"DDDD";

wait for 10ns; des\_a1 <= '0'; des\_a2 <= '1'; des\_a3 <= '1'; data <= x"CCCC";

wait for 10ns; des\_a1 <= '1'; des\_a2 <= '0'; des\_a3 <= '0'; data <= x"BBBB";

wait for 10ns; des\_a1 <= '1'; des\_a2 <= '0'; des\_a3 <= '1'; data <= x"AAAA";

wait for 10ns; des\_a1 <= '1'; des\_a2 <= '1'; des\_a3 <= '0'; data <= x"9999";

wait for 10ns; des\_a1 <= '1';des\_a2 <= '1';des\_a3 <= '1';data <= x"8888";

end process;

END;