Mux code

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

use ieee.numeric_std.all;

entity mux4ta1 is --this is the code that used for the 4 to 1 mux
port( a,b,c,d: in std_logic_vector(3 downto 0);
sel: in std_logic_vector(1 downto 0);
z: out std_logic_vector (3 downto 0));

end mux4ta1;

architecture mux of mux4ta1 is
begin
process( a,b,c,d,sel) is
begin
--this part of the code is going to dicatate what happens if select bits are picked
if (sel(0) ='0' and sel(1) = '0') then
     z <= a;
  elsif (sel(0) ='1' and sel(1) = '0') then
     z <= c;
  elsif (sel(0) ='0' and sel(1) = '1') then
     z <= b;
  else
     z <= d;
  end if;
end process;
end mux;
```

mux top level code

```
--This part of the code is used for the top level of the mux
entity muxcomp is
port(a,b: in bit_vector (3 downto 0);
sel : in bit_vector (1 downto 0);
z:out bit_vector (3 downto 0));
end;
--This part of the code makes the mux

architecture muxcomp2 of muxcomp is
component mux
port( a,b,c,d: in bit_vector(3 downto 0);
sel: in bit_vector(1 downto 0);
z: out bit_vector (3 downto 0));
end component;
--This part of the code makes the mux a component
signal op0:bit_vector (3 downto 0);
signal op1:bit_vector (3 downto 0);
signal op2:bit_vector (3 downto 0);
signal op3:bit_vector (3 downto 0);
--This part of the code controls all the operations needed for the top level
begin
op0<=((a(3) and b(3))&  (a(2) and b(2))& (a(1) and b(1))&(b(0) and a(0)));
op1<=
((a(3) or b(3))&  (a(2) or b(2))& (a(1) or b(1))&(b(0) or a(0)));
op2<=
((a(3) nand b(3))&  (a(2) nand b(2))& (a(1) nand b(1))&(b(0) nand a(0)));
op3<=
((a(3) nor b(3))&  (a(2) nor b(2))& (a(1) nor b(1))&(b(0) nor a(0)));
        --This part of the code says all the operations that need tob e used
muxco2:mux port map(op0,op1,op2,op3,sel,z);
--This part of the code puts all of them into a port map so they can be used for later
end;
```

Test bench for top level

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

use ieee.numeric_std.all;

entity test_muxcomp is

end test_muxcomp;

architecture test of test_muxcomp is

signal a,b: std_logic_vector(3 downto 0);

signal sel: std_logic_vector(1 downto 0);

signal z: std_logic_vector(3 downto 0);
```

```
component mux is

port( a,b,c,d : in std_logic_vector(3 downto 0);
sel: in std_logic_vector (1 downto 0);
y : out std_logic_vector(3 downto 0) );

end component;



--below are all my test symbols that will be used
 signal ta,tb,tc,td: std_logic_vector(3 downto 0);
signal selT: std_logic_vector(1 downto 0);
signal y: std_logic_vector(3 downto 0);

begin

DUT1:mux  port map(ta,tb,tc,td,selT,y);

process

begin



--this is to toggle the select bits so that I can get different outputs and that is done for steps below it
selT <= "00";
wait for 10 ns;
selT <= "01";
wait for 10 ns;
selT <="10";
```

```
wait for 10 ns;

selT <="11";

wait for 10 ns;

end process;


process

begin

a <="1001";

b<= "1100";



wait for 100 ns;



--report "input = " & std_logic'image(input(2))   & std_logic'image(input(1))  & std_logic'image(input(0))
& " isprime = " & std_logic'image(isprime);



end process;end test;
```

Test bench for mux

```
library ieee;


use ieee.std_logic_1164.all;


use ieee.std_logic_unsigned.all;


use ieee.numeric_std.all;


entity test_mux is
```

```
end test_mux;

architecture test of test_mux is

signal a,b: std_logic_vector(3 downto 0);

signal sel: std_logic_vector(1 downto 0);

signal z: std_logic_vector(3 downto 0);

component mux is

port( a,b,c,d : in std_logic_vector(3 downto 0);
sel: in std_logic_vector (1 downto 0);
y : out std_logic_vector(3 downto 0) );

end component;



 signal ta,tb,tc,td: std_logic_vector(3 downto 0);
signal selT: std_logic_vector(1 downto 0);
signal y: std_logic_vector(3 downto 0);

begin

DUT1:mux  port map(ta,tb,tc,td,selT,y);
```

```
process

begin



--this is to toggle the select bits so that I can get different outputs and that is done for steps below it


selT <= "00";

wait for 10 ns;

selT <= "01";

wait for 10 ns;

selT <="10";

wait for 10 ns;

selT <="11";

wait for 10 ns;

end process;


process

begin


for i in 0 to 15 loop

ta <= std_logic_vector(to_unsigned(i,4));

tb <= std_logic_vector(to_unsigned(i,4));

tc <= std_logic_vector(to_unsigned(i,4));

td <= std_logic_vector(to_unsigned(i,4));


wait for 100 ns;
```

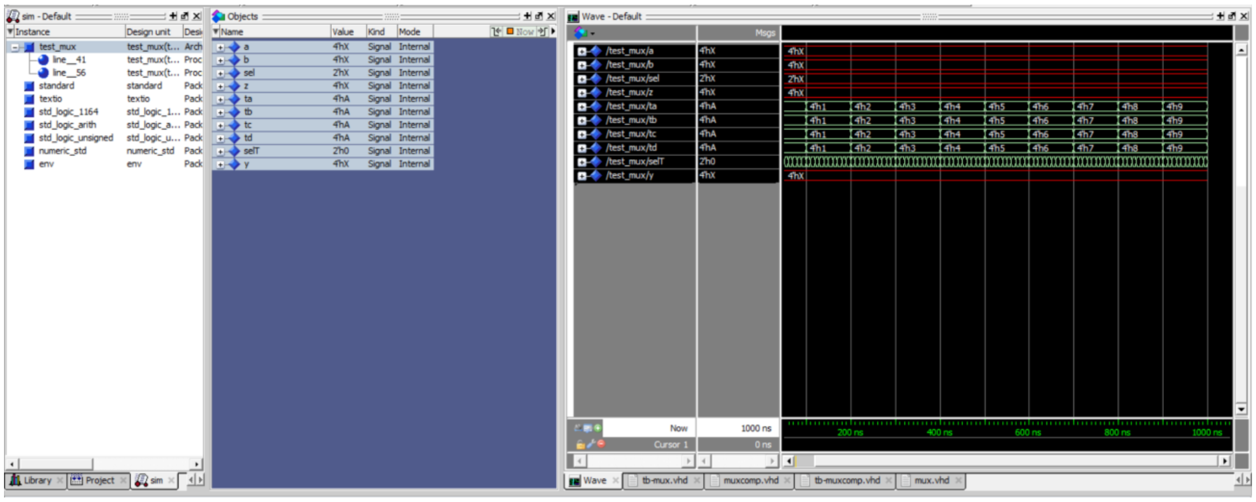
```
--report "input = " & std_logic'image(input(2))   & std_logic'image(input(1))  & std_logic'image(input(0)) & " isprime = " & std_logic'image(isprime);

assert y /= y;

end loop;

std.env.stop(0);

end process;end test;
```



sim - Default

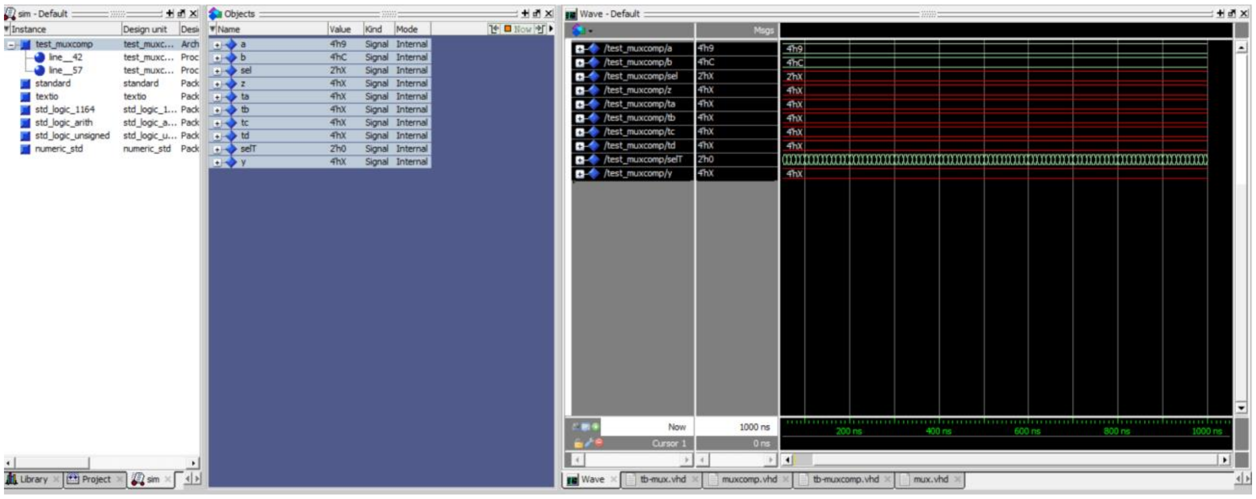
Instance	Design unit	Desi
test_mux	test_mux(t...	Arch
line__41	test_mux(t...	Proc
line__56	test_mux(t...	Proc
standard	standard	Pack
textio	textio	Pack
std_logic_1164	std_logic_1...	Pack
std_logic_arith	std_logic_a...	Pack
std_logic_unsigned	std_logic_u...	Pack
numeric_std	numeric_std	Pack
env	env	Pack

Objects

Name	Value	Kind	Mode
a	4'hX	Signal	Internal
b	4'hX	Signal	Internal
sel	2'hX	Signal	Internal
z	4'hX	Signal	Internal
ta	4'hA	Signal	Internal
tb	4'hA	Signal	Internal
tc	4'hA	Signal	Internal
td	4'hA	Signal	Internal
selT	2'h0	Signal	Internal
y	4'hX	Signal	Internal

Wave - Default

	Msgs
/test_mux/a	4'hX
/test_mux/b	4'hX
/test_mux/sel	2'hX
/test_mux/z	4'hX
/test_mux/ta	4'hA
/test_mux/tb	4'hA
/test_mux/tc	4'hA
/test_mux/td	4'hA
/test_mux/selT	2'h0
/test_mux/y	4'hX



sim - Default

Instance	Design unit	Desi
test_muxcomp	test_muxc...	Arch
line__42	test_muxc...	Proc
line__57	test_muxc...	Proc
standard	standard	Pack
textio	textio	Pack
std_logic_1164	std_logic_1...	Pack
std_logic_arith	std_logic_a...	Pack
std_logic_unsigned	std_logic_u...	Pack
numeric_std	numeric_std	Pack

Objects

Name	Value	Kind	Mode
a	4'h9	Signal	Internal
b	4'hC	Signal	Internal
sel	2'hX	Signal	Internal
z	4'hX	Signal	Internal
ta	4'hX	Signal	Internal
tb	4'hX	Signal	Internal
tc	4'hX	Signal	Internal
td	4'hX	Signal	Internal
selT	2'h0	Signal	Internal
y	4'hX	Signal	Internal

Wave - Default

	Msgs
/test_muxcomp/a	4'h9
/test_muxcomp/b	4'hC
/test_muxcomp/sel	2'hX
/test_muxcomp/z	4'hX
/test_muxcomp/ta	4'hX
/test_muxcomp/tb	4'hX
/test_muxcomp/tc	4'hX
/test_muxcomp/td	4'hX
/test_muxcomp/selT	2'h0
/test_muxcomp/y	4'hX