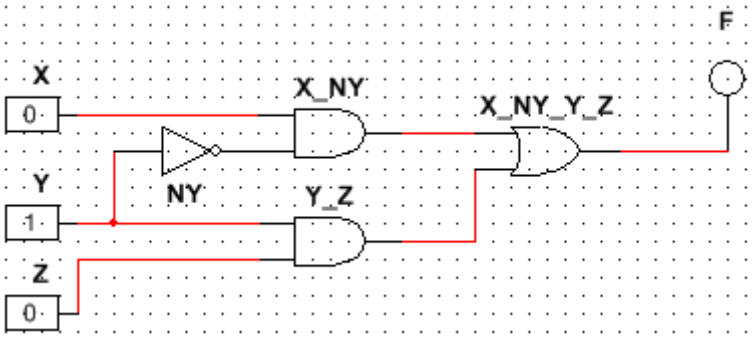
Lab 3

1) Truth Table:

Truth Table			
X	Y	Z	F1(XY' + YZ)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

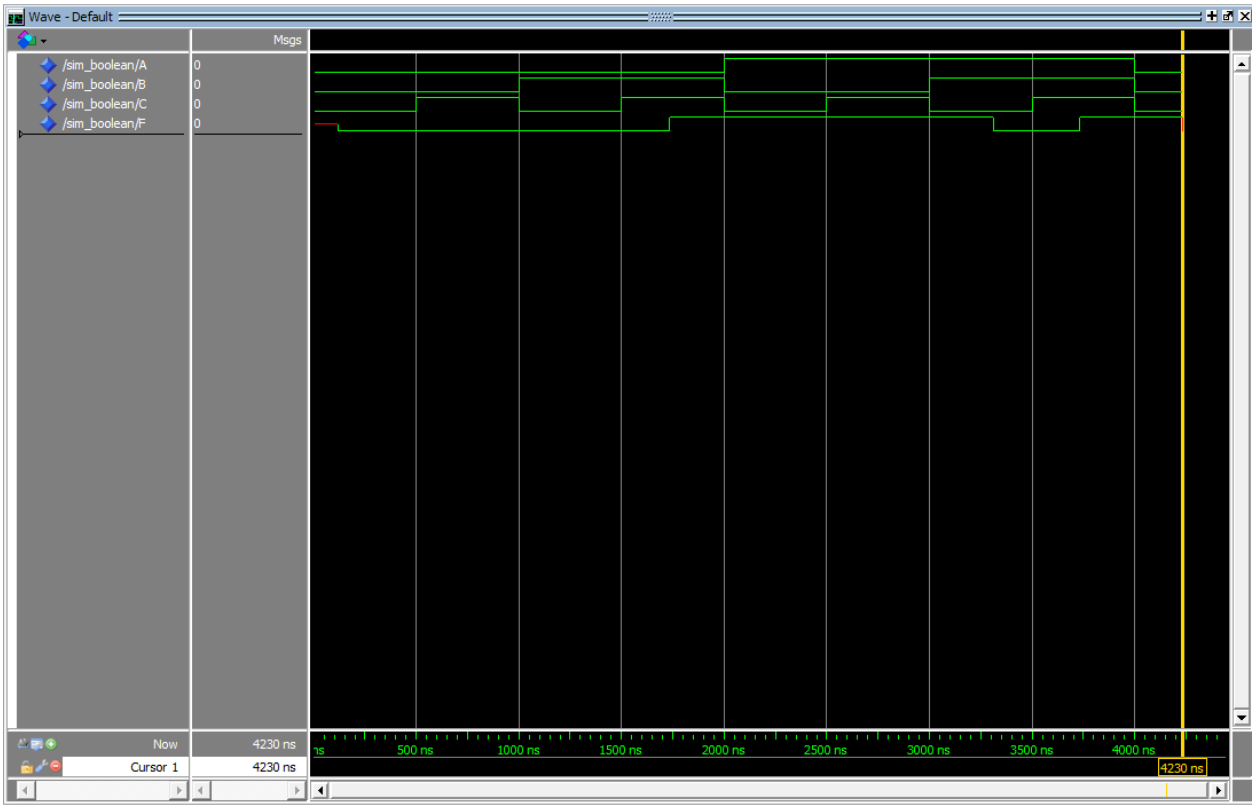
2) Dataflow Style:

a. Circuit:



b. Dataflow Style:

i. Boolean Style:



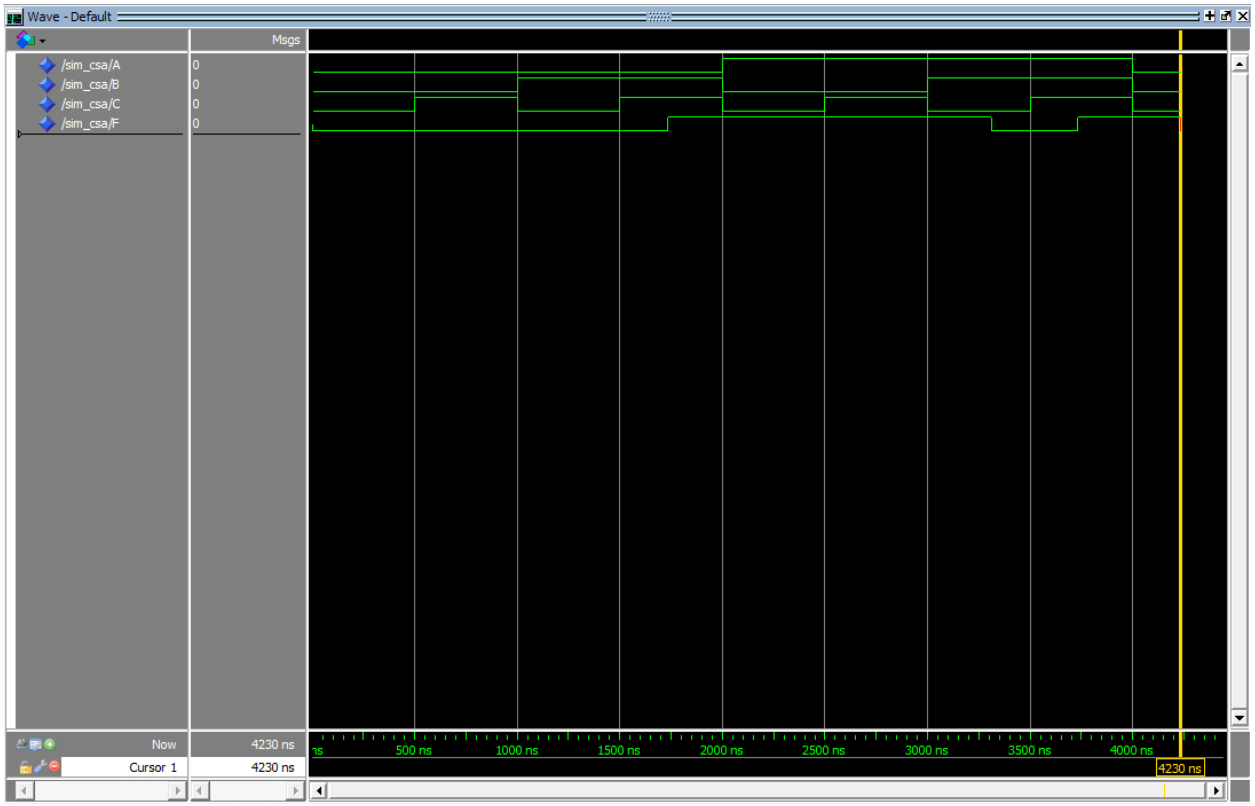
Boolean Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Cory_Boolean is port (
X, Y, Z : in STD_LOGIC;
F : out STD_LOGIC
);
end Cory_Boolean;

architecture Long_Dataflow_Boolean of Cory_Boolean is
signal NY, X_NY, Y_Z : std_logic := '0';
begin
        NY <= (Not Y) after 80 ns;
        X_NY <= (X and NY) after 110 ns;
        Y_Z <= (Y and Z) after 110 ns;
        F <= (X_NY or Y_Z) after 120 ns;
end Long_Dataflow_Boolean;
-- Begin of test bench section
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- test bench name is sim
-- In modelsim right click on "sim" to simulate
-- In modelsim right click on "sim" to add wave
ENTITY sim_Boolean IS
END sim_Boolean;
ARCHITECTURE behavior OF sim_Boolean IS
--Component Declaration for the Unit Under Test (UUT)
COMPONENT Cory_Boolean
PORT(
X : IN std_logic;
Y : IN std_logic;
Z : IN std_logic;
F : OUT std_logic
);
END COMPONENT;
-- Test bench stimuli signals to be generated
-- to drive the input signals of the design under test
-- in this case the DUT is the 3-input AND gate
signal A : std_logic := '0';
signal B : std_logic := '0';
signal C : std_logic := '0';
-- Test bench signals to represent the output
signal F : std_logic;
-- begin of test bench action
BEGIN
--Instantiate the Unit Under Test (UUT)
-- and connect stimuli signals of the test bench
-- to the UUT pins.
uut: Cory_Boolean PORT MAP (
X => A,
Y => B,
Z => C,
F => F
);
-- Stimulus Process example
-- we use 500 ns pulse width for this example
```

```
-- you can change the stimuli pulse width
-- to test your design accordingly
process
begin
--stimulus inputs
A <= '0'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '0'; wait;
end process;
end;
```

ii. CSA Style:



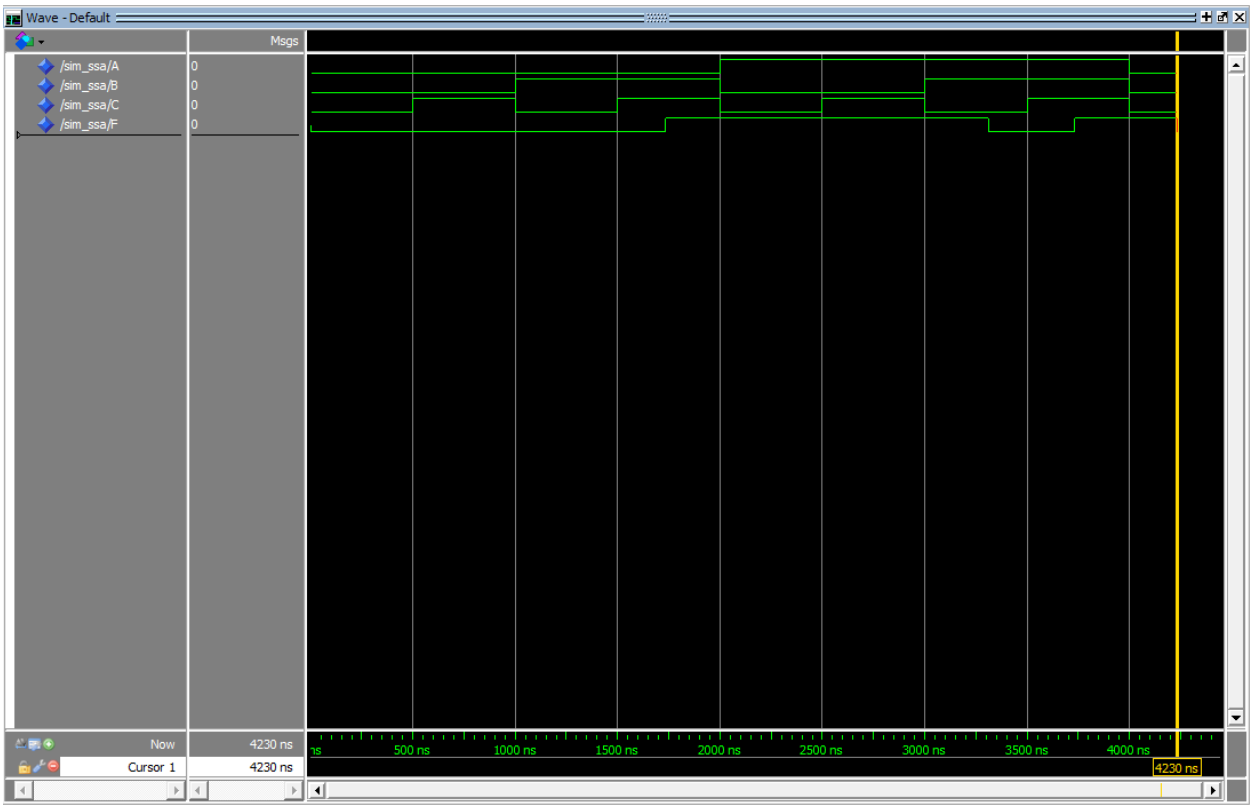
CSA Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Cory_CSA is port (
X, Y, Z : in STD_LOGIC;
F : out STD_LOGIC
);
end Cory_CSA;

architecture Long_Dataflow_CSA of Cory_CSA is
signal NY, X_NY, Y_Z, Q : std_logic := '0';
begin
        NY <= (Not Y) after 80 ns;
        X_NY <= (X and NY) after 110 ns;
        Y_Z <= (Y and Z) after 110 ns;
        Q <= (X_NY or Y_Z) after 120 ns;
        F <= '1' when Q = '1'
        else '0';
end Long_Dataflow_CSA;
```

```
-- Begin of test bench section
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- test bench name is sim
-- In modelsim right click on "sim" to simulate
-- In modelsim right click on "sim" to add wave
ENTITY sim_CSA IS
END sim_CSA;
ARCHITECTURE behavior OF sim_CSA IS
--Component Declaration for the Unit Under Test (UUT)
COMPONENT Cory_CSA
PORT(
X : IN std_logic;
Y : IN std_logic;
Z : IN std_logic;
F : OUT std_logic
);
END COMPONENT;
-- Test bench stimuli signals to be generated
-- to drive the input signals of the design under test
-- in this case the DUT is the 3-input AND gate
signal A : std_logic := '0';
signal B : std_logic := '0';
signal C : std_logic := '0';
-- Test bench signals to represent the output
signal F : std_logic;
-- begin of test bench action
BEGIN
--Instantiate the Unit Under Test (UUT)
-- and connect stimuli signals of the test bench
-- to the UUT pins.
uut: Cory_CSA PORT MAP (
X => A,
Y => B,
Z => C,
F => F
);
-- Stimulus Process example
```

```
-- we use 500 ns pulse width for this example
-- you can change the stimuli pulse width
-- to test your design accordingly
process
begin
--stimulus inputs
A <= '0'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '0'; wait;
end process;
end;
```

iii. SSA Style:



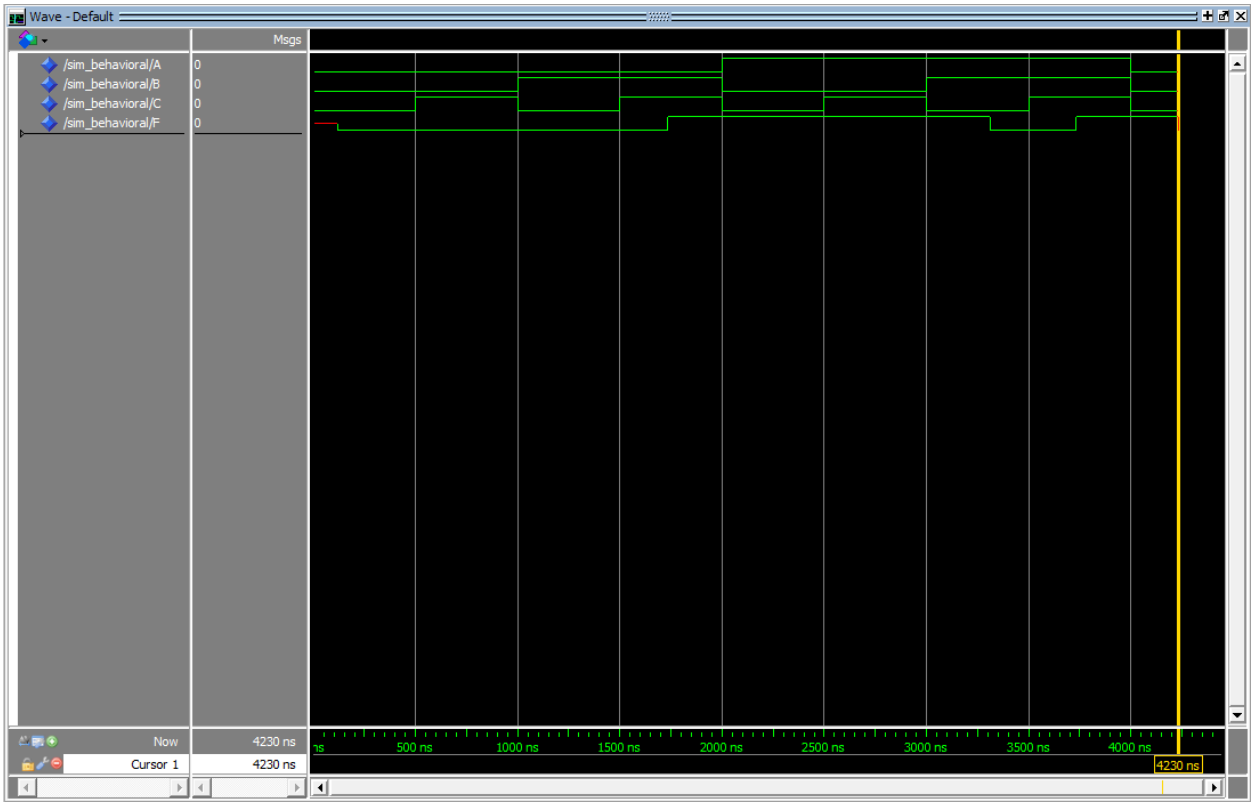
SSA Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Cory_SSA is port (
X, Y, Z : in STD_LOGIC;
F : out STD_LOGIC
);
end Cory_SSA;

architecture Long_Dataflow_SSA of Cory_SSA is
signal NY, X_NY, Y_Z, Q : std_logic := '0';
begin
        NY <= (Not Y) after 80 ns;
        X_NY <= (X and NY) after 110 ns;
        Y_Z <= (Y and Z) after 110 ns;
        Q <= (X_NY or Y_Z) after 120 ns;
        with Q select
        F <= '1' when '1',
            '0' when '0',
```

```
                '0' when others;
end Long_Dataflow_SSA;
-- Begin of test bench section
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- test bench name is sim
-- In modelsim right click on "sim" to simulate
-- In modelsim right click on "sim" to add wave
ENTITY sim_SSA IS
END sim_SSA;
ARCHITECTURE behavior OF sim_SSA IS
--Component Declaration for the Unit Under Test (UUT)
COMPONENT Cory_SSA
PORT(
X : IN std_logic;
Y : IN std_logic;
Z : IN std_logic;
F : OUT std_logic
);
END COMPONENT;
-- Test bench stimuli signals to be generated
-- to drive the input signals of the design under test
-- in this case the DUT is the 3-input AND gate
signal A : std_logic := '0';
signal B : std_logic := '0';
signal C : std_logic := '0';
-- Test bench signals to represent the output
signal F : std_logic;
-- begin of test bench action
BEGIN
--Instantiate the Unit Under Test (UUT)
-- and connect stimuli signals of the test bench
-- to the UUT pins.
uut: Cory_SSA PORT MAP (
X => A,
Y => B,
Z => C,
F => F
```

```
);
-- Stimulus Process example
-- we use 500 ns pulse width for this example
-- you can change the stimuli pulse width
-- to test your design accordingly
process
begin
--stimulus inputs
A <= '0'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '0'; wait;
end process;
end;
```

3) Behavioral Style:



Behavioral Code:

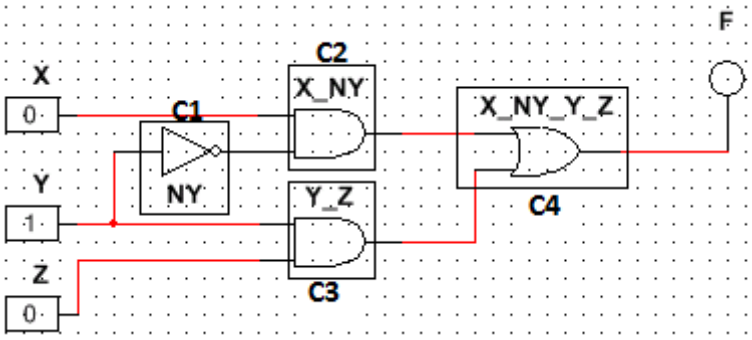
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Cory_Behavioral is port (
X, Y, Z : in STD_LOGIC;
F : out STD_LOGIC
);
end Cory_Behavioral;

architecture Long_Behavioral of Cory_Behavioral is
signal NY, X_NY, Y_Z : std_logic := '0';
begin
sequential: process (X, Y, Z, NY, X_NY, Y_Z)
begin
        NY <= (Not Y) after 80 ns;
        X_NY <= (X and NY) after 110 ns;
        Y_Z <= (Y and Z) after 110 ns;
        F <= (X_NY or Y_Z) after 120 ns;
end process sequential;
```

```
end Long_Behavioral;
-- Begin of test bench section
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- test bench name is sim
-- In modelsim right click on "sim" to simulate
-- In modelsim right click on "sim" to add wave
ENTITY sim_Behavioral IS
END sim_Behavioral;
ARCHITECTURE behavior OF sim_Behavioral IS
--Component Declaration for the Unit Under Test (UUT)
COMPONENT Cory_Behavioral
PORT(
X : IN std_logic;
Y : IN std_logic;
Z : IN std_logic;
F : OUT std_logic
);
END COMPONENT;
-- Test bench stimuli signals to be generated
-- to drive the input signals of the design under test
-- in this case the DUT is the 3-input AND gate
signal A : std_logic := '0';
signal B : std_logic := '0';
signal C : std_logic := '0';
-- Test bench signals to represent the output
signal F : std_logic;
-- begin of test bench action
BEGIN
--Instantiate the Unit Under Test (UUT)
-- and connect stimuli signals of the test bench
-- to the UUT pins.
uut: Cory_Behavioral PORT MAP (
X => A,
Y => B,
Z => C,
F => F
);
```

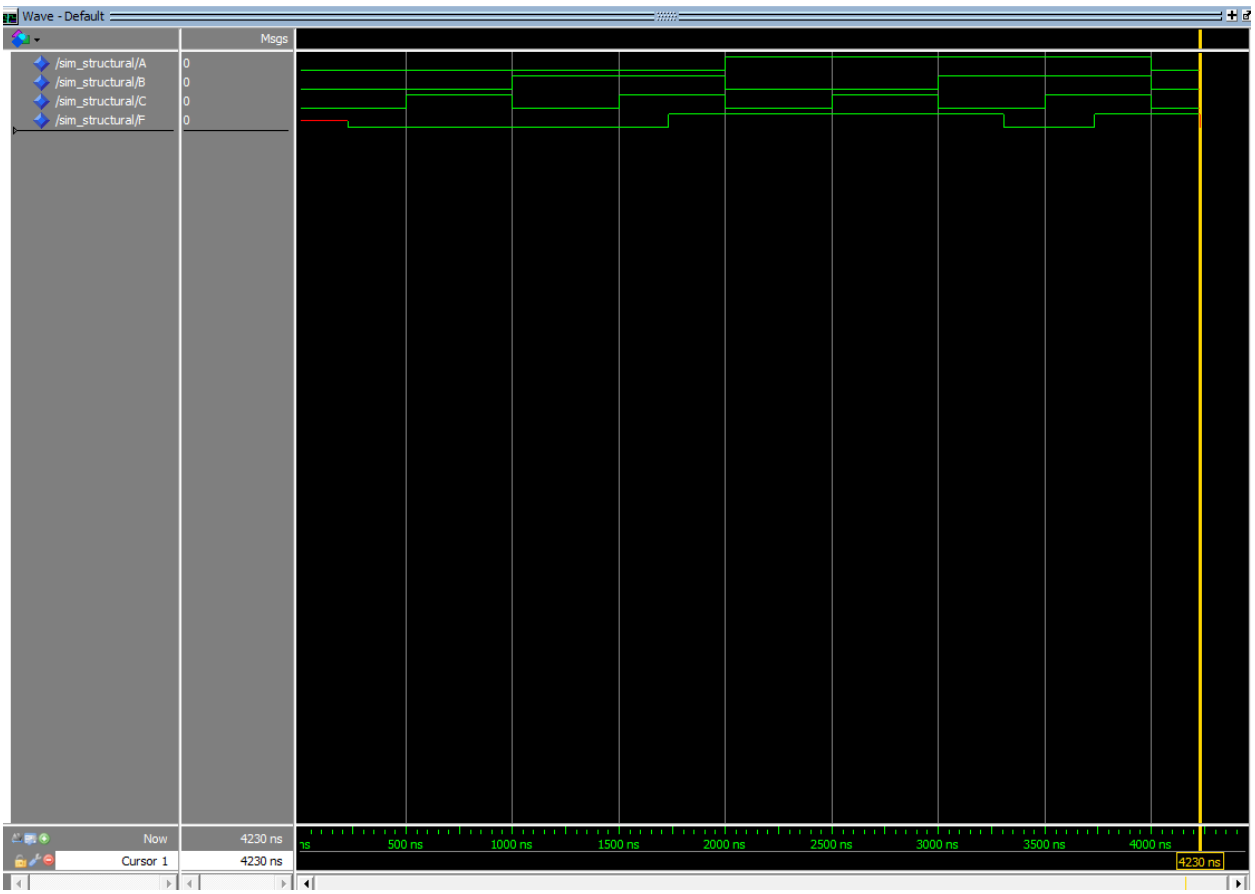
```
-- Stimulus Process example
-- we use 500 ns pulse width for this example
-- you can change the stimuli pulse width
-- to test your design accordingly
process
begin
--stimulus inputs
A <= '0'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '0'; wait;
end process;
end;
```

4) VHDL Structural Style:

a. Structural Circuit:



b. Structural Style:



Structural Code:

```
--Component definition for OR_1
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Cory_Not_1 is port (
i1: in std_logic;
o1: out std_logic
);
end Cory_Not_1;
architecture Long_Not_1 of Cory_Not_1 is
begin
o1 <= (not i1) after 80 ns;
end Long_Not_1;

--Component definition for AND_1
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Cory_And_1 is port (
```

```
i1, i2: in std_logic;
o1: out std_logic
);
end Cory_And_1;
architecture Long_And_1 of Cory_And_1 is
begin
o1 <= (i1 and i2) after 110 ns;
end Long_And_1;

--Component definition for AND_2
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Cory_And_2 is port (
i1, i2: in std_logic;
o1: out std_logic
);
end Cory_And_2;
architecture Long_And_2 of Cory_And_2 is
begin
o1 <= (i1 and i2) after 110 ns;
end Long_And_2;

--Component definition for OR_1
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Cory_Or_1 is port (
i1, i2: in std_logic;
o1: out std_logic
);
end Cory_Or_1;
architecture Long_Or_1 of Cory_Or_1 is
begin
o1 <= (i1 or i2) after 120 ns;
end Long_Or_1;

--Structural Design (Top Level)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity Cory_Structural is port (
X, Y, Z : in STD_LOGIC;
F : out STD_LOGIC
);
end Cory_Structural;
architecture Long_Structural of Cory_Structural is
        signal NY, X_NY, Y_Z : std_logic := '0';
component Cory_Not_1 is port (
        i1: in std_logic;
        o1: out std_logic
        );
end component;

component Cory_And_1 is port (
        i1, i2: in std_logic;
        o1: out std_logic
        );
end component;

component Cory_And_2 is port (
        i1, i2: in std_logic;
        o1: out std_logic
        );
end component;

component Cory_Or_1 is port (
        i1, i2: in std_logic;
        o1: out std_logic
        );
end component;

begin
        C1: Cory_Not_1 port map (i1 => Y, o1 => NY);
        C2: Cory_And_1 port map (i1 => X, i2 => NY, o1 => X_NY);
        C3: Cory_And_2 port map (i1 => Y, i2 => Z, o1 => Y_Z);
        C4: Cory_Or_1 port map (i1 => X_NY, i2 => Y_Z, o1 => F);
end Long_Structural;
```

```
-- Begin of test bench section
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- test bench name is sim
-- In modelsim right click on "sim" to simulate
-- In modelsim right click on "sim" to add wave
ENTITY sim_Structural IS
END sim_Structural;
ARCHITECTURE behavior OF sim_Structural IS
--Component Declaration for the Unit Under Test (UUT)
COMPONENT Cory_Structural
PORT(
X : IN std_logic;
Y : IN std_logic;
Z : IN std_logic;
F : OUT std_logic
);
END COMPONENT;
-- Test bench stimuli signals to be generated
-- to drive the input signals of the design under test
-- in this case the DUT is the 3-input AND gate
signal A : std_logic := '0';
signal B : std_logic := '0';
signal C : std_logic := '0';
-- Test bench signals to represent the output
signal F : std_logic;
-- begin of test bench action
BEGIN
--Instantiate the Unit Under Test (UUT)
-- and connect stimuli signals of the test bench
-- to the UUT pins.
uut: Cory_Structural PORT MAP (
X => A,
Y => B,
Z => C,
F => F
);
-- Stimulus Process example
```

```
-- we use 500 ns pulse width for this example
-- you can change the stimuli pulse width
-- to test your design accordingly
process
begin
--stimulus inputs
A <= '0'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '0'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '0'; C <= '1'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '0'; wait for 500 ns;
A <= '1'; B <= '1'; C <= '1'; wait for 500 ns;
A <= '0'; B <= '0'; C <= '0'; wait;
end process;
end;
```

5) Conclusions:

This project was a daunting task at first. It was difficult to get started and hard to follow along with the book. After more reading into the chapter about the differences between the design styles, it became easier. The different styles start to lead into the features of the next one. After doing the first few, it was easy to understand what was being implemented by the designs.

After doing this lab I have a much better understanding of the operation of modelsim and the ability to successfully use the test bench, which is much easier than forcing the wave signs.