Names: John Doe, Jane Smith
Login: jdoe, jsmith
Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
Lab Section: 0##
Assignment: Lab #
I acknowledge all content is original.

I. Lab objective
The lab objective was to introduce us to using Xilinx ISE 13 tool-chain.

II . Personal contributions
The work was shared for this exercise. As a group we followed the steps to create a Xilinx project, synthesize, test, and place & route the design. John Doe actually created the files on the computer, while Jane Smith aided in following steps to complete the lab.

III. Skill learned & knowledge gained.
Not having worked with the Xilinx tool-chain before, I learned how to create a project, add new vhdl source files, create a testbench, simulate VHDL source code, synthesize,and place and route. The tool-chain most difficult part was running the simulation because the first time the program did not operate correctly. After some trouble-shooting we found the simulation process was active but not accessing the correct file. To fix the error we killed all processes to Xilinx and the reloaded the project. Overlooking the code, I learned how to make a concurrent and sequential statements within VHDL, as well as a simple testbench. The features of Xilinx GUI made programming in VHDL easy and I still want to explore the other tools within the tool-chain. After reading the Xilinx documentation, I understand many of the other features such as core generator and language templates.

IV. Known bugs locations
We tested our design with all possible cases, therefore are no known bugs.

V. Feedback on the lab
Overall the lab was great!

```
--Names: John Doe, Jane Smith
--Login: jdoe, jsmith
--Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
--Lab Section: 0##
--Assignment: Lab #
--I acknowledge all content is original.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity half_subtractor is
    Port ( X : in  STD_LOGIC;
           Y : in  STD_LOGIC;
           D : out  STD_LOGIC;
           B : out  STD_LOGIC);
end half_subtractor;

architecture Behavioral of half_subtractor is

begin

D <= X XOR Y;
B <= NOT(X) AND Y;

end Behavioral;
```

```vhdl
--Names: John Doe, Jane Smith
--Login: jdoe, jsmith
--Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
--Lab Section: 0##
--Assignment: Lab #
--I acknowledge all content is original.

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY half_subtractor_tb IS
END half_subtractor_tb;

ARCHITECTURE behavior OF half_subtractor_tb IS

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

    COMPONENT half_subtractor
    PORT(
         X : IN  std_logic;
         Y : IN  std_logic;
         D : OUT  std_logic;
         B : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal X : std_logic := '0';
   signal Y : std_logic := '0';

        --Outputs
   signal D : std_logic;
   signal B : std_logic;

        --Clock
        signal clock : std_logic;
   constant clock_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: half_subtractor PORT MAP (
          X => X,
          Y => Y,
          D => D,
          B => B
        );

   -- Clock process definitions
   clock_process :process
   begin
                clock <= '0';
                wait for clock_period/2;
                clock <= '1';
                wait for clock_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
                --Hold reset state
            X <= '0';
            Y <= '0';
      wait for clock_period*10;

      -- insert stimulus here
            --Cases
            X <= '0';
            Y <= '0';
            wait for clock_period;
            Assert ( D = '0' and B = '0' )
```



```vhdl
                  Report "Case 0"
                  Severity ERROR;

            X <= '0';
            Y <= '1';
            wait for clock_period;
            Assert ( D = '1' and B = '1' )
                  Report "Case 1"
                  Severity ERROR;

                                    X <= '1';
            Y <= '0';
            wait for clock_period;
            Assert ( D = '1' and B = '0' )
                  Report "Case 2"
                  Severity ERROR;

                                    X <= '1';
            Y <= '1';
            wait for clock_period;
            Assert ( D = '0' and B = '0' )
                  Report "Case 3"
                  Severity ERROR;

                Report "Done with testbench." severity NOTE;
      wait;
   end process;

END;
```

```
--Names: John Doe, Jane Smith
--Login: jdoe, jsmith
--Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
--Lab Section: 0##
--Assignment: Lab #
--I acknowledge all content is original.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_subtractor is
    Port ( BI : in  STD_LOGIC;
           X : in  STD_LOGIC;
           Y : in  STD_LOGIC;
           D : out  STD_LOGIC;
           B : out  STD_LOGIC);
end full_subtractor;

architecture Behavioral of full_subtractor is

begin

D <= X XOR Y XOR BI;
B <= (BI AND NOT(X XOR Y)) OR (NOT(X) AND Y);

end Behavioral;
```

```vhdl
--Names: John Doe, Jane Smith
--Login: jdoe, jsmith
--Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
--Lab Section: 0##
--Assignment: Lab #
--I acknowledge all content is original.

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY full_subtractor_tb IS
END full_subtractor_tb;

ARCHITECTURE behavior OF full_subtractor_tb IS

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

    COMPONENT full_subtractor
    PORT(
         BI : IN  std_logic;
         X : IN  std_logic;
         Y : IN  std_logic;
         D : OUT  std_logic;
         B : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal BI : std_logic := '0';
   signal X : std_logic := '0';
   signal Y : std_logic := '0';

        --Outputs
   signal D : std_logic;
   signal B : std_logic;

   --Clock
        signal clock : std_logic;
   constant clock_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: full_subtractor PORT MAP (
          BI => BI,
          X => X,
          Y => Y,
          D => D,
          B => B
        );

   -- Clock process definitions
   clock_process :process
   begin
                clock <= '0';
                wait for clock_period/2;
                clock <= '1';
                wait for clock_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
--Hold reset state
            X <= '0';
            Y <= '0';
                                BI <= '0';
      wait for clock_period*10;

      -- insert stimulus here
            --Cases
```



```vhdl
            X <= '0';
            Y <= '0';
                                BI <= '0';
            wait for clock_period;
            Assert ( D = '0' and B = '0' )
                  Report "Case 0"
                  Severity ERROR;

            X <= '0';
            Y <= '0';
                                BI <= '1';
            wait for clock_period;
            Assert ( D = '1' and B = '1' )
                  Report "Case 1"
                  Severity ERROR;

                                X <= '0';
            Y <= '1';
                                BI <= '0';
            wait for clock_period;
            Assert ( D = '1' and B = '1' )
                  Report "Case 2"
                  Severity ERROR;

                                X <= '0';
            Y <= '1';
                                BI <= '1';
            wait for clock_period;
            Assert ( D = '0' and B = '1' )
                  Report "Case 3"
                  Severity ERROR;

                                X <= '1';
            Y <= '0';
                                BI <= '0';
            wait for clock_period;
            Assert ( D = '1' and B = '0' )
                  Report "Case 4"
                  Severity ERROR;

                                X <= '1';
            Y <= '0';
                                BI <= '1';
            wait for clock_period;
            Assert ( D = '0' and B = '0' )
                  Report "Case 5"
                  Severity ERROR;

                                X <= '1';
            Y <= '1';
                                BI <= '0';
            wait for clock_period;
            Assert ( D = '0' and B = '0' )
                  Report "Case 6"
                  Severity ERROR;

                                X <= '1';
            Y <= '1';
                                BI <= '1';
            wait for clock_period;
            Assert ( D = '1' and B = '1' )
                  Report "Case 7"
                  Severity ERROR;

                Report "Done with testbench." severity NOTE;
      wait;
   end process;

END;
```

```vhdl
--Names: John Doe, Jane Smith
--Login: jdoe, jsmith
--Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
--Lab Section: 0##
--Assignment: Lab #
--I acknowledge all content is original.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity generic_subtractor is
    Generic ( WIDTH : INTEGER := 32);
    Port ( X : in  STD_LOGIC_VECTOR (WIDTH-1 downto 0);
           Y : in  STD_LOGIC_VECTOR (WIDTH-1 downto 0);
           D : out  STD_LOGIC_VECTOR (WIDTH-1 downto 0);
           B : out  STD_LOGIC);
end generic_subtractor;

architecture Structural of generic_subtractor is

component half_subtractor
    Port ( X : in  STD_LOGIC;
           Y : in  STD_LOGIC;
           D : out  STD_LOGIC;
           B : out  STD_LOGIC);
end component half_subtractor;

component full_subtractor
    Port ( BI : in  STD_LOGIC;
           X : in  STD_LOGIC;
           Y : in  STD_LOGIC;
           D : out  STD_LOGIC;
           B : out  STD_LOGIC);
end component full_subtractor;

signal B_sig : STD_LOGIC_VECTOR(WIDTH-1 downto 0);

        begin

                start : half_subtractor
                        port map (
                                X => X(0),
                                Y => Y(0),
                                D => D(0),
                                B => B_sig(0)
                                );

                gen_sub : for N in 1 to WIDTH-1 generate
                        sub_unit : full_subtractor
                                port map (
                                          BI => B_sig(N-1),
                                          X => X(N),
                                          Y => Y(N),
                                          D => D(N),
                                          B => B_sig(N)
                                        );
                end generate gen_sub;

                B <= B_sig(WIDTH-1);

end Structural;
```

```vhdl
--Names: John Doe, Jane Smith
--Login: jdoe, jsmith
--Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
--Lab Section: 0##
--Assignment: Lab #
--I acknowledge all content is original.

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY generic_subtractor_tb IS
END generic_subtractor_tb;

ARCHITECTURE behavior OF generic_subtractor_tb IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT generic_subtractor
    PORT(
         X : IN  std_logic_vector(31 downto 0);
         Y : IN  std_logic_vector(31 downto 0);
         D : OUT  std_logic_vector(31 downto 0);
         B : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal X : std_logic_vector(31 downto 0) := (others => '0');
   signal Y : std_logic_vector(31 downto 0) := (others => '0');

        --Outputs
   signal D : std_logic_vector(31 downto 0);
   signal B : std_logic;

        --Clock
        signal clock : std_logic;
   constant clock_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: generic_subtractor PORT MAP (
          X => X,
          Y => Y,
          D => D,
          B => B
        );

   -- Clock process definitions
   clock_process :process
   begin
                clock <= '0';
                wait for clock_period/2;
                clock <= '1';
                wait for clock_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 10 ns.
                X <= (others => '0');
                Y <= (others => '0');
      wait for 10 ns;

      -- insert stimulus here

                --Error Check Min
                X <= X"00000001";
                        --Set input
                Y <= conv_std_logic_vector(1,32);
--Set input
```



```vhdl
                wait for clock_period;
        --Wait for gate delay
                --If not correct then report Error
                Assert ( D = conv_std_logic_vector(0,32) and
                                        B = '0')
                                --Output check
                        Report "Error at Min."
        --Report to command line
                        Severity ERROR;
                --Type of severity to report

                --Error Check underflow 1
                X <= X"00000000";
                        --Set input
                Y <= conv_std_logic_vector(1,32);
--Set input
                wait for clock_period;
        --Wait for gate delay
                --If not correct then report Error
                Assert ( D = X"FFFFFFFF" and
                                        B = '1' )
                                --Output check
                        Report "Error at Underflow 1."
--Report to command line
                        Severity ERROR;
                --Type of severity to report

                --Error Check underflow 2
                X <= X"00000000";
                        --Set input
                Y <= X"FFFFFFFF";
                        --Set input
                wait for clock_period;
        --Wait for gate delay
                --If not correct then report Error
                Assert ( D = X"00000001"
                                        and B = '1' )
                        --Output check
                        Report "Error at Underflow 2."
--Report to command line
                        Severity ERROR;
                --Type of severity to report

                --Error Check for ranged subtractions
                for N in 1024 downto 512 loop
                        for M in 512 downto 0 loop
                                X <= conv_std_logic_vector(N,32);
        --Set input to N with 32 bit range
                                Y <= conv_std_logic_vector(M,32);
        --Set input to M with 32 bit range
                                wait for clock_period;
                        --Wait for gate delay
                                --If not correct then report Error
                                Assert ( D = conv_std_logic_vector(N-M,32) and
                                                B = '0' )
                                        --Output check with N+M value
                                        Report "Error at" &
                                                " A=" & integer'image(N
) &                     --Print strings and N,M, N+M to
                                                " B=" & integer'image(M
) &                     --  counsole upon error.
                                                " S=" & integer'image(N
-M) &
                                                "."--Report to command
line
                                        Severity ERROR;
                                --Type of severity to report
                        end loop;
                end loop;
        --Systems runs for (10ns)*(3+(N+1)*(M+1))= ? ns

        Report "Done with testbench" Severity NOTE;                     --Report
 testbench complete.
```

```
    wait;
    end process;

END;
```

```c
/*
Names: John Doe, Jane Smith
Login: jdoe, jsmith
Email: jdoe@cs.ucr.edu, jsmith@cs.ucr.edu
Lab Section: 0##
Assignment: Lab #
I acknowledge all content is original.
*/

#include <stdio.h>

int main()
{

        printf("%s\n", "An example C file.");
        return 0;
}

```

```
# To generate the turnin, type "make all".
# To run other commands type "make <command>". I.E. "make turnin".
# To remove the executeable and any extraneous files type "make clean" at
# the command prompt.
###########################################################################
#Variable to hold lab number
LABNUM=0

#Variable to hold first lastname
LASTNAME1=doe

#Variable to hold second lastname
LASTNAME2=smith

#Variable of Tarball generated
TAR=lab$(LABNUM)_$(LASTNAME1)_$(LASTNAME2).tgz

#Variable of Post Script generated
PSPRINT=lab$(LABNUM)_$(LASTNAME1)_$(LASTNAME2).ps
PDFPRINT=lab$(LABNUM)_$(LASTNAME1)_$(LASTNAME2).pdf

#File hiearachy is lab$(LABNUM)_$(PART)$(SUBPART)$(TESTBENCH).$(EXTENSION)
#$(LABNUM)= lab number
#$(PART)= if a lab part exists in order from 1 to infinity
#$(SUBPART)= if other components were used to make this component that were
#            not included then put them here in order from a to z. If "a"
#            requires sub parts then put "a"$(subsubpart) I.E. "a1" now
#            numbering. Continue letter/number trend to infinity.
#$(TESTBENCH)= if this is a test bench file then put "tb" to indicate it
#              is test bench for this part
#$(EXTENSION)= is the character file extension

#Files to be included in the PS/PDF print
FILES=lab0_report.txt half_subtractor.vhdl half_subtractor_tb.vhdl full_subtractor.vhdl full_subtractor_tb.vhdl generic_subtractor.vhdl generic_subtractor_tb.vhdl gen.c Makefile

#Variable of to use G++ GNU compiler to compile C files
COMPILER=gcc

# -g Produce debug information, necessary for debugging.
# -Wall         Show all reasonable warnings...
# -Werror Make all warnings into errors.
# -ansi This flag tells the compiler to enforce ANSI C standards
# -pedantic More pedantic ansi, warnings for stuff you probably didn't mean.
# -O2 Optimize even more. GCC performs nearly all supported optimizations that
#  do not involve a space-speed tradeoff. As compared to -O, this option
#       increases both compilation time and the performance of the generated code.
# -O3 Optimize yet more. -O3 turns on all optimizations specified by -O2
#  and also turns on the -finline-functions, -funswitch-loops,
#  -fpredictive-commoning, -fgcse-after-reload, -ftree-vectorize and
#  -fipa-cp-clone options.

#Variable of GCC Compiler flags
COMPILEFLAGS= -g -Wall -Werror -ansi -pedantic -O3

#Variable of Program name
PROGRAM=genvhdl

#Targets
#Compile all, specifically first clean, print,
# turnin, and compile in this order
all: clean print turnin

#Compile C code
compile:
	$(COMPILER) $(COMPILEFLAGS) -o $(PROGRAM) gen.c

#Remove unnecessary files
clean:
	rm -rf *~ $(PSPRINT) $(PDFPRINT) $(TAR) $(PROGRAM)

```



```
#Print files to a .ps document
print:
	a2ps -M letter --line-numbers=1 -o $(PSPRINT) $(FILES)

printpdf:
	enscript -2 --fancy-header --line-numbers=1 --truncate-lines \
	--word-wrap --style=emacs --tabsize=3 --landscape $(FILES) \
	-o - | ps2pdfwr - $(PDFPRINT)

#Create a tar ball for project turn in
turnin:
	tar -czvf $(TAR) $(PSPRINT) $(FILES)
```