EECE 320 HOMEWORK 3

**Problem 1**

* Table 5-38

library ieee;

use ieee.std\_logic\_1164.all;

entity prime is

 port (N: in std\_logic\_vector (3 downto 0);

 F: out std\_logic);

end prime;

architecture prime3\_arch of prime is

signal N3L\_N0, N3L\_N2L\_N1, N2L\_N1\_N0, N2\_N1L\_N0: STD\_LOGIC;

begin

 N3L\_N0 <= '1' when N(3)='0' and N(0)='1' else '0';

 N3L\_N2L\_N1 <= '1' when N(3)='0' and N(2)='0' and N(1)='1' else '0';

 N2L\_N1\_N0 <= '1' when N(2)='0' and N(1)='1' and N(0)='1' else '0';

 N2\_N1L\_N0 <= '1' when N(2)='1' and N(1)='0' and N(0)='1' else '0';

 F<= N3L\_N0 or N3L\_N2L\_N1 or N2L\_N1\_N0 or N2\_N1L\_N0;

end prime3\_arch;

* Table 5-40

library ieee;

use ieee.std\_logic\_1164.all;

entity prime is

 port (N: in st\_logic\_vector (3 downto 0);

 F: out std\_logic);

end prime;

architecture prime4\_arch of prime is

begin

 with N select

 F <= '1' when "0001",

 '1' when "0010",

 '1' when "0011" | "0101" | "0111",

 '1' when "1011" | "1101",

 '0' when others;

end prime4\_arch;

* Table 5-45 (function from table 5-25)

library ieee;

use ieee.std\_logic\_1164.all;

entity prime is

 port (N: in std\_logic\_vector (3 downto 0);

 F: out std\_logic);

end prime;

architecture prime7\_arch of prime is

function CONV\_INTEGER (X: STD\_LOGIC\_VECTOR) return INTEGER is

 variable RESULT: INTEGER;

 begin

 RESULT := 0;

 for i in X'range loop

 RESULT := RESULT\*2;

 case X(i) is

 when '0' | 'L' => null;

 when '1' | 'H' => RESULT := RESULT+1;

 when others => null;

 end case;

 end loop;

 return RESULT;

end CONV\_INTEGER;

begin

 process(N)

 variable NI: INTEGER;

 begin

 NI := CONV\_INTEGER(N);

 if NI=1 or NI=2 then F <= '1';

 elsif NI=3 or NI=5 or NI=7 or NI=11 or NI=13 then F <= '1';

 else F <= '0';

 end if;

 end process;

end prime7\_arch;

* Test Bench for the three tables:

library ieee;

use ieee.std\_logic\_1164.all;

entity prime\_tb is

end prime\_tb;

architecture arch\_tb of prime\_tb is

 component prime port (N: in std\_logic\_vector (3 downto 0);

 F: out std\_logic);

 end component;

signal N: std\_logic\_vector (3 downto 0) := "0000";

signal F: std\_logic;

begin

 test: prime port map(N=>N, F=>F);

 process

 begin

 N<="0000"; wait for 10 ns;

 N<="0001"; wait for 10 ns;

 N<="0010"; wait for 10 ns;

 N<="0011"; wait for 10 ns;

 N<="0100"; wait for 10 ns;

 N<="0101"; wait for 10 ns;

 N<="0110"; wait for 10 ns;

 N<="0111"; wait for 10 ns;

 N<="1000"; wait for 10 ns;

 N<="1001"; wait for 10 ns;

 N<="1010"; wait for 10 ns;

 N<="1011"; wait for 10 ns;

 N<="1100"; wait for 10 ns;

 N<="1101"; wait for 10 ns;

 N<="1110"; wait for 10 ns;

 N<="1111";

 wait;

 end process;

end arch\_tb;

* Waveform for the three codes:

The testing simulation waveform is the same for all three codes, the wave in yellow is the output that detects prime numbers (=1 when X=1, 2, 3, 5, 7, 11, and 13):

**Problem 2**

* Circuit and expression:

The 2-bit comparator minimal sum-of-products circuit that produces a 1 output if P < Q is:

Q0

P0

P1

Q1

And its logic expression is: F = P1'.Q1 + P0'.Q1.Q0 + P1'. P0'.Q0

* Entity and data flow architecture:

The VHDL entity and data flow architecture that describes the circuit in terms of gates are:

library ieee;

use ieee.std\_logic\_1164.all;

entity compare is

 port(P: in std\_logic\_vector(1 downto 0);

 Q: in std\_logic\_vector(1 downto 0);

 F: out std\_logic);

end compare;

architecture compare\_arch of compare is

 signal s1,s2,s3:std\_logic;

begin

 s1<= ((not P(1)) and Q(1)) ;

 s2<= ((not P(0)) and Q(1) and Q(0));

 s3<= ((not P(1)) and (not P(0)) and Q(0));

 F<= s1 or s2 or s3;

end compare\_arch;

* Test Bench:

library ieee;

use ieee.std\_logic\_1164.all;

entity compare\_tb is

end compare\_tb;

architecture arch\_tb of compare\_tb is

component compare port (P,Q: in std\_logic\_vector (1 downto 0);

 F: out std\_logic);

end component;

signal P,Q: std\_logic\_vector (1 downto 0);

signal F: std\_logic;

begin

testBench : compare port map (P=>P, Q=>Q, F=>F);

process

begin

P<="00"; Q<="00"; wait for 10 ns;

Q<="01"; wait for 10 ns;

Q<="10"; wait for 10 ns;

Q<="11"; wait for 10 ns;

P<="01"; Q<="00"; wait for 10 ns;

Q<="01"; wait for 10 ns;

Q<="10"; wait for 10 ns;

Q<="11"; wait for 10 ns;

P<="10"; Q<="00"; wait for 10 ns;

Q<="01"; wait for 10 ns;

Q<="10"; wait for 10 ns;

Q<="11"; wait for 10 ns;

P<="11"; Q<="00"; wait for 10 ns;

Q<="01"; wait for 10 ns;

Q<="10"; wait for 10 ns;

Q<="11"; wait for 10 ns;

end process;

end arch\_tb;

* VHDL entity and behavioral architecture:

library ieee;

use ieee.std\_logic\_1164.all;

entity compare is

 port (P,Q: in std\_logic\_vector (1 downto 0);

 F: out std\_logic);

end compare;

architecture behav\_arch of compare is

 signal s1,s2,s3,s4: std\_logic;

 begin

 process (P,Q)

 begin

 if ( (P="00" and Q= "01") or (P="00" and Q(1)='1')

 or (P="01" and Q(1)= '1') or (P="10" and Q="11") ) then F<='1';

 else F<= '0';

 end if;

 end process;

end behav\_arch;

The test bench is the same as for the dataflow architecture.

* Simulation waveform:

The testing simulation waveform is the same for both architectures, the output is equal to 1 when P < Q:

**Problem 3**

* VHDL entity and architecture:

library ieee;

use ieee.std\_logic\_1164.all;

entity multiple is

 port (N: in std\_logic\_vector (5 downto 0);

 M3: out std\_logic;

 M5: out std\_logic);

end multiple;

architecture multiple\_arch of multiple is

function CONV\_INTEGER (X: STD\_LOGIC\_VECTOR) return INTEGER is

 variable RESULT: INTEGER;

 begin

 RESULT := 0;

 for i in X'range loop

 RESULT := RESULT\*2;

 case X(i) is

 when '0' | 'L' => null;

 when '1' | 'H' => RESULT := RESULT+1;

 when others => null;

 end case;

 end loop;

 return RESULT;

end CONV\_INTEGER;

begin

 process(N)

 variable temp: integer;

 begin

 temp := CONV\_INTEGER(N);

 if (temp rem 3)=0 then M3<='1';

 else M3 <= '0';

 end if;

 if (temp rem 5)=0 then M5<='1';

 else M5 <= '0';

 end if;

 end process;

end multiple\_arch;

* Test Bench:

use ieee.std\_logic\_1164.all;

entity multiple\_tb is

end multiple\_tb;

architecture multiple\_tb\_arch of multiple\_tb is

component multiple port (N: in std\_logic\_vector (5 downto 0);

 M3: out std\_logic;

 M5: out std\_logic);

end component;

signal N: std\_logic\_vector (5 downto 0);

signal M3,M5: std\_logic;

begin

test: multiple port map (N,M3,M5);

 process

 begin

 N<="000110"; wait for 10 ns; -- N=6

 N<="001001"; wait for 10 ns; -- N=9

 N<="001111"; wait for 10 ns; -- N=15

 N<="010100"; wait for 10 ns; -- N=20

 N<="010111"; wait for 10 ns; -- N=23

 N<="101100"; wait for 10 ns; -- N=44

 end process;

 end multiple\_tb\_arch;

* Simulation waveform:

The wave on top indicates whether the integer is a multiple of 3, and the one on the bottom indicates whether it’s a multiple of 5.