# Appedix

## [1] Gates.VHDL

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity MYAND is

port (A , B : in std\_logic ;

Q : out std\_logic );

End ;

ARCHitecture AND1 OF MYAND IS

BEGIN

Q<= A AND B AFTER 4 NS ;

END ;

------------------------------

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity MYOR is

port (A , B : in std\_logic ;

Q: out std\_logic );

End ;

ARCHitecture OR1 OF MYOR IS

BEGIN

Q<= A OR B AFTER 4 NS ;

END ;

---------------------------------

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity MYXOR is

port (A , B : in std\_logic ;

Q: out std\_logic );

End ;

ARCHitecture XOR1 OF MYXOR IS

BEGIN

Q<= A XOR B AFTER 7 NS ;

END ;

--------------------------------

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity MYXNOR is

port (A , B : in std\_logic ;

Q: out std\_logic );

End ;

ARCHitecture XNOR1 OF MYXNOR IS

BEGIN

Q<= A XNOR B AFTER 6 NS ;

END ;

---------------------------------

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity MYNOR is

port (A , B : in std\_logic ;

Q: out std\_logic );

End ;

ARCHitecture NOR1 OF MYNOR IS

BEGIN

Q<= A NOR B AFTER 3 NS ;

END ;

---------------------------------

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity MYNAND is

port (A , B : in std\_logic ;

Q: out std\_logic );

End ;

ARCHitecture NAND1 OF MYNAND IS

BEGIN

Q<= A NAND B AFTER 3 NS ;

END ;

---------------------------------

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity MYINVERTER is

port (A : in std\_logic ;

Q: out std\_logic );

End ;

ARCHitecture INVERT1 OF MYINVERTER IS

BEGIN

Q<= NOT A AFTER 2 NS ;

END ;

---------------------------------

## [2] Equality.VHDL

Library Ieee ;

use ieee.std\_logic\_1164.all ;

-- entity for the equality Circuit

entity AEQUALB is

PORT ( a , b : IN std\_logic\_vector(7 downto 0) ;

C: OUT std\_logic ;

e: out std\_logic\_Vector(7 downto 0 )); -- e is an output because i will use it in the next circuit

end entity ;

-- Architecture for the equality

Architecture AEQUALB1 of AEQUALB is

SIGNAL h,f : STD\_logic\_VEctor (7 DOWNTO 0 ) ;

begin

-- Generation for all xnor gates in the equality circuit XNOR checks equality

XNORS: for i in 0 to 7 generate

XNOR1 : entity work.MYXNOR(XNOR1) PORT MAP (a(i),b(i),h(i));

end generate XNORS ;

-- the and gates in the equality circuit

and7 :entity work.MYAND(AND1) port map (h(7) , h(6) , f(6)) ;

and6 :entity work.MYAND(AND1) port map (f(6) , h(5) , f(5)) ;

and5 :entity work.MYAND(AND1) port map (f(5) , h(4) , f(4)) ;

and4 :entity work.MYAND(AND1) port map (f(4) , h(3) , f(3)) ;

and3 :entity work.MYAND(AND1) port map (f(3) , h(2) , f(2)) ;

and2 :entity work.MYAND(AND1) port map (f(2) , h(1) , f(1)) ;

and1 :entity work.MYAND(AND1) port map (f(1) , h(0) , f(0)) ;

-- driving the final output

f(7) <= h(7) ;

e<= f ;

C <=f(0);

End architecture AEQUALB1;

## [3] notequal.VHDL

LIBRARY IEEE ;

Use Ieee.std\_logic\_1164.all;

-- entity for the a>b or a<b Circuit

Entity ANOTEQUALB is

PORT ( A , B : IN STD\_LOGIc\_vector(7 DOWNTO 0 ) ;

S : IN STD\_logic;

e : IN STd\_logic\_VEctor(7 DOWNTO 0) ; --A=B

ABIGGER , ASMALLER : OUT STD\_logic ); -- A>B , A<B

END ENTITY ;

ARCHItecture COMPARE OF ANOTEQUALB IS

SIGNAL AS , BS : STD\_logic:= '0' ; -- COMPARING THE SIGN

SIGNAL q : STD\_logic\_vector(7 DOWNTO 0 );

SIGNal Y : STD\_logic\_vector(7 DOWNTO 0 ) ;

SIGNal L : STD\_logic\_vector(7 DOWNTO 0 ) ;

SIGNAL LB : STD\_logic\_vector(7 DOWNTO 0) ; -- A COPY OF B

BEGIN

-- I WILL FOLLOW THE GATES SHOWN IN THE STRUCTURE SUBMITTED WITH THE PROJECT REPORT WITH THE SAME NAMES

NOTB : FOR I IN 0 TO 6 GENErate -- TAKING THE NEGATION FOR B

not1: entity work.MYINVERTER(INVERT1) PORT MAP(B(I) , LB(I));

END GENerate NOTB ;

LB(7) <= B(7) ; -- last bit is not negated

ANDS : for i in 0 to 6 generate -- a and not b loop

ands: entity work.MYAND(AND1) PORT MAP (A(i) , LB(i) ,L(i) ) ;

END GENERate ANDS ;

ANDS2 : for i in 0 to 6 generate

ands2: entity work.MYAND(AND1) PORT MAP (e(i+1) , L(I) ,q(i) ) ;

END GENERate ANDS2 ;

-- the following 3 lines compare the sign of a and b

xor1 : ENTITy WORK.MYXOR(XOR1) PORT MAP (A(7) , s , AS) ;

xnor1: ENTITy WORK.MYXNOR(XNOR1) PORT MAP (B(7) , s , BS) ;

AND2 : ENTITy WORK.MYAND(AND1) PORT MAP (AS , BS , q(7)) ;

-- THE FOLLOWING LINES ARE THE REMAINING GATES IN THE STRUCTURE

OR1 : ENTity WORK.MYOR(OR1) PORT MAP (q(7) , q(6), Y(3)) ;

OR2 : ENTity WORK.MYOR(OR1) PORT MAP (q(5) , q(4) , Y(2)) ;

OR3 : ENTity WORK.MYOR(OR1) PORT MAP (q(3) , q(2) , Y(1)) ;

OR4 : ENTity WORK.MYOR(OR1) PORT MAP (q(1) , q(0) , Y(0 )) ;

OR5 : ENTity WORK.MYOR(OR1) PORT MAP (Y(0) , Y(1) , Y(4)) ;

OR6 : ENTity WORK.MYOR(OR1) PORT MAP (Y(2), Y(3) ,Y(5)) ;

OR7 : ENTity WORK.MYOR(OR1) PORT MAP (Y(4) , Y(5) , Y(6)) ;

NOR1 : ENTITy WORK.MYNOR(NOR1) PORT MAP (e(0) , Y(6) , y(7)); -- A is smaller than b if not bigger nor equal to it

ASMALLER <= Y(7) ;

ABIGGER <= Y(6) ;

END ARCHitecture ;

## [4] final assembly.vhdl

-- this entity will assemble both a=b and a>b circuits together to get the final comparator

Library IEEE ;

USE IEEE.STD\_LOGIC\_1164.ALL ;

ENTITY COMPARATOR IS

PORT ( a,b : in std\_logic\_vector (7 downto 0) ;

s ,clk : in std\_logic ;

Greater, Smaller , EQUAL : out std\_logic ) ;

End Entity ;

Architecture compare of COMPARATOR is

Signal e : std\_logic\_vector (7 downto 0 ) ;

signal lGreater , lSmaller , lEqual : std\_logic ;

signal la , lb : std\_logic\_vector(7 downto 0);

Begin

equality : Entity work.AEQUALB(AEQUALB1) port map ( la , lb , lEQUAL , e) ;

comparison : Entity work.ANOTEQUALB(COMPARE) port map (la,lb,s, e ,lGreater , lSmaller) ;

process (clk)

begin

if (rising\_edge(clk) ) then

la <= a ;

lb<=b;

Greater <= lGreater ;

Smaller <= lSmaller ;

Equal <= lEqual ;

end if ;

End process ;

End architecture ;

## [5] final assembly with testing.vhdl

-- this entity has the generator , analyser , and coparator together

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL ;

entity fulltest is

end ;

architecture testing of fulltest is

signal clock , expectedequal , expectedgreater ,expectedsmaller , realequal , realgreater , realsmaller , s : std\_logic ;

signal atest,btest : std\_logic\_vector (7 downto 0 );

begin

generator : Entity work.Generator1(generation) port map (clock , atest , btest , expectedequal , expectedgreater , expectedsmaller ) ;

comparator : Entity work.COMPARATOR(compare) port map (atest,btest , s , clock , realgreater , realsmaller , realequal) ;

analyser : Entity work.Analyzer(tb) port map ( expectedgreater , expectedsmaller , expectedequal , realgreater , realsmaller , realequal , clock) ;

end ;

## [6]Value Generator

-- This entity will generate values for the testing operation

Library IEEE ;

USE IEEE.STd\_logic\_1164.ALL ;

use IEEE.std\_logic\_arith.all;

Entity Generator1 is

port ( clock : in std\_logic ;

atest , btest : out std\_logic\_vector(7 downto 0 ) ;

expectedequal , expectedgreater , expectedsmaller : out std\_logic );

end ;

ARCHitecture generation of Generator1 is

signal ga, gb : std\_logic\_vector(7 downto 0 );

signal lexpectedequal , lexpectedgreater , lexpectedsmaller : std\_logic ;

begin

PROCESS

BEGIN

FOR I IN 0 TO 255 LOOP

FOR J IN 0 TO 255 LOOP

-- Set the inputs to the adder

ga <= conv\_std\_logic\_vector(i,8);

gb <= CONV\_STD\_LOGIC\_VECTOR(j,8);

atest<= ga;

btest<=gb ;

-- Calculate what the output of the adder should be

if (ga>gb) then

expectedgreater <= '1' after 60 ns ;

expectedsmaller<= '0' after 60 ns;

expectedequal <= '0' after 60 ns;

elsif(ga<gb) then

expectedgreater <= '0' after 60 ns;

expectedsmaller<= '1' after 60 ns;

expectedequal <= '0' after 60 ns;

elsif(ga=gb) then

expectedgreater <= '0' after 60 ns;

expectedsmaller<= '1' after 60 ns;

expectedequal <= '0' after 60 ns;

end if ;

-- Wait until adder output has settled

WAIT until rising\_edge(clock);

END LOOP;

END LOOP;

WAIT;

END PROCESS;

end ;

## [7]Analyzer.Vhdl

-- this entity will check if the expected value is the same as the output of the comparator

Library IEEE ;

USE IEEE.STD\_logic\_1164.ALL;

ENTity Analyzer is

port ( expectedG ,expectedS , expectedE , G , S ,E , clock: IN STD\_logic );

end ;

ARCHITECTURE tb OF Analyzer Is

BEGIN

PROCESS(clock)

BEGIN

IF rising\_edge(clock) THEN

-- Check whether comparator output matches expectation

ASSERT expectedG = G and expectedS = S and expectedE = E

REPORT "Comparison Faild"

SEVERITY WARNING;

END IF;

END PROCESS;

END ARCHITECTURE tb;