LAB REPORT

COEN 313

Digital Systems Design II

Lab Section: AJ-X

Experiment #:2: Structural and Concurrent VHDL

Mathias Desrochers

ID: 40241734

Date Performed: May 21, 2024

Date Due: May 28, 2024

OBJECTIVES

- Acquaintance with structural and concurrent VHDL.
- Familiarization with different VHDL coding styles.
- Understanding of combinational logic minimization using logic synthesis tools.

QUESTIONS

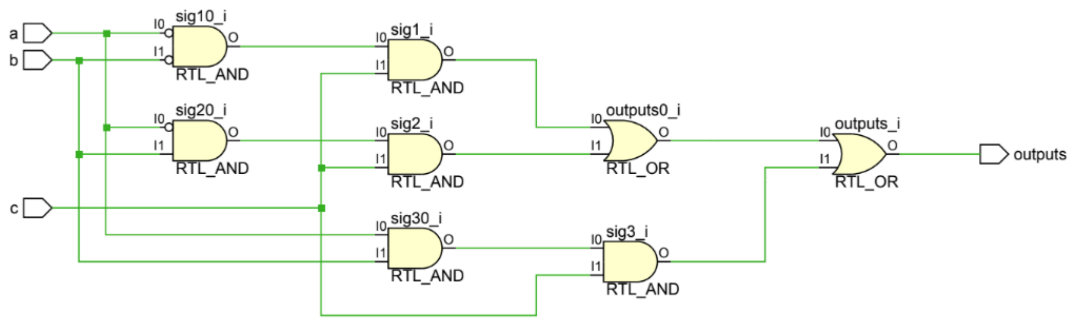
1. Rewrite the VHDL code for the sum_of_minterms entity making use of only CSA statements (no port maps). Re-synthesize your design with Vivado and compare the resulting RTL elaborated and Implemented schematic diagrams with that of the original design. Comment on any differences/similarities among the schematics. You do not have to download this version of the circuit to the FPGA board.

Answers :

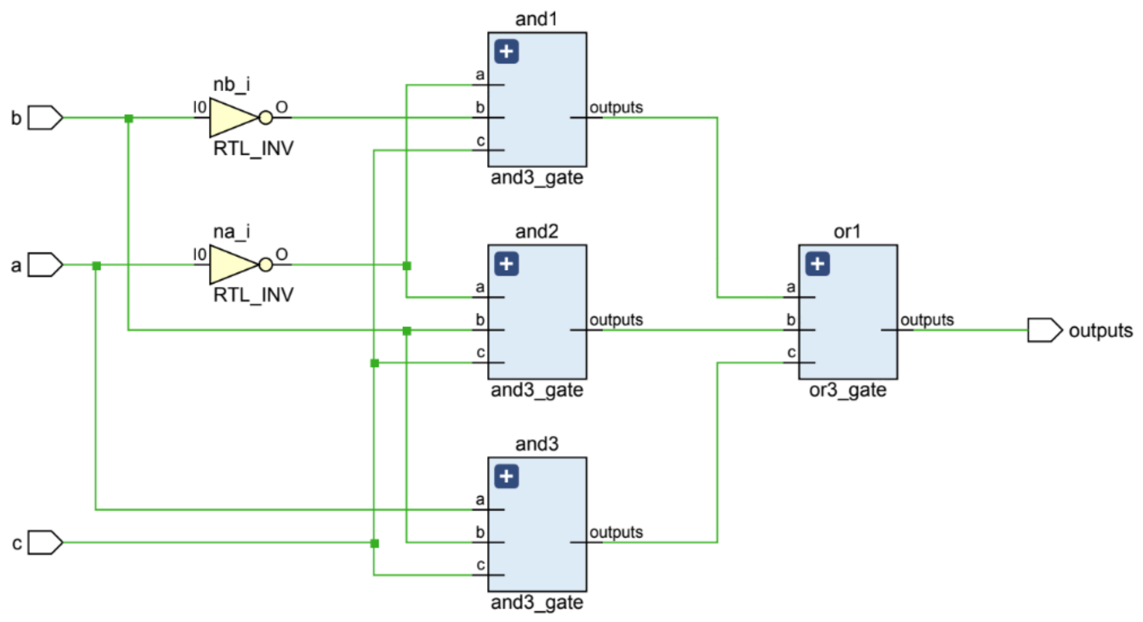
```
library ieee;
use ieee.std_logic_1164.all;

entity sums is
  port(a, b, c : in std_logic;
       outputs : out std_logic);
end sums;




architecture structural of sums is


signal na ,nb,nc : std_logic;
signal sig1,sig2,sig3     : std_logic;

begin
```

```
-- component instantiation
na <= not a;
nb <= not b;
nc <= not c;
sig1 <= (na and nb) and c;
sig2 <= (na and b) and c;
sig3 <= (a and b) and c;
outputs <= (sig1 or sig2) or sig3;

end structural
```

using port :



With port :



We can clearly see the complexity of the system without the port is harder to understand has more gate are shown, as such if you were to have a huge diagram it would be hard to understand.
We can also see that only 2 input gates can be done without port while with port we can simulate a higher input gate. That said they both result in the same output.

2. Determine the Boolean function which the LUT implements in both versions of the VHDL

code. Is the function equal to the original sum-of-minterms expression described by the VHDL

Code?

LUT - > look up table
Yes, both are the same, both equation are out = (not_a and not_b and c) or (not_a and b and c) or (a and b and c)

3. Do the two RTL elaborated schematics indicate whether the synthesis tool has performed any

logic minimization?

RTL -> Register-Transfer Level

Compare both transistor level implementation on the vivado
Answer:

In both situation the tool seem to have performed simplification since they look similar, even thought with port we have a higher level

5. Determine whether the following VHDL code results in RTL elaborated and Implemented

schematics which are indicative of logic minimization having been performed:

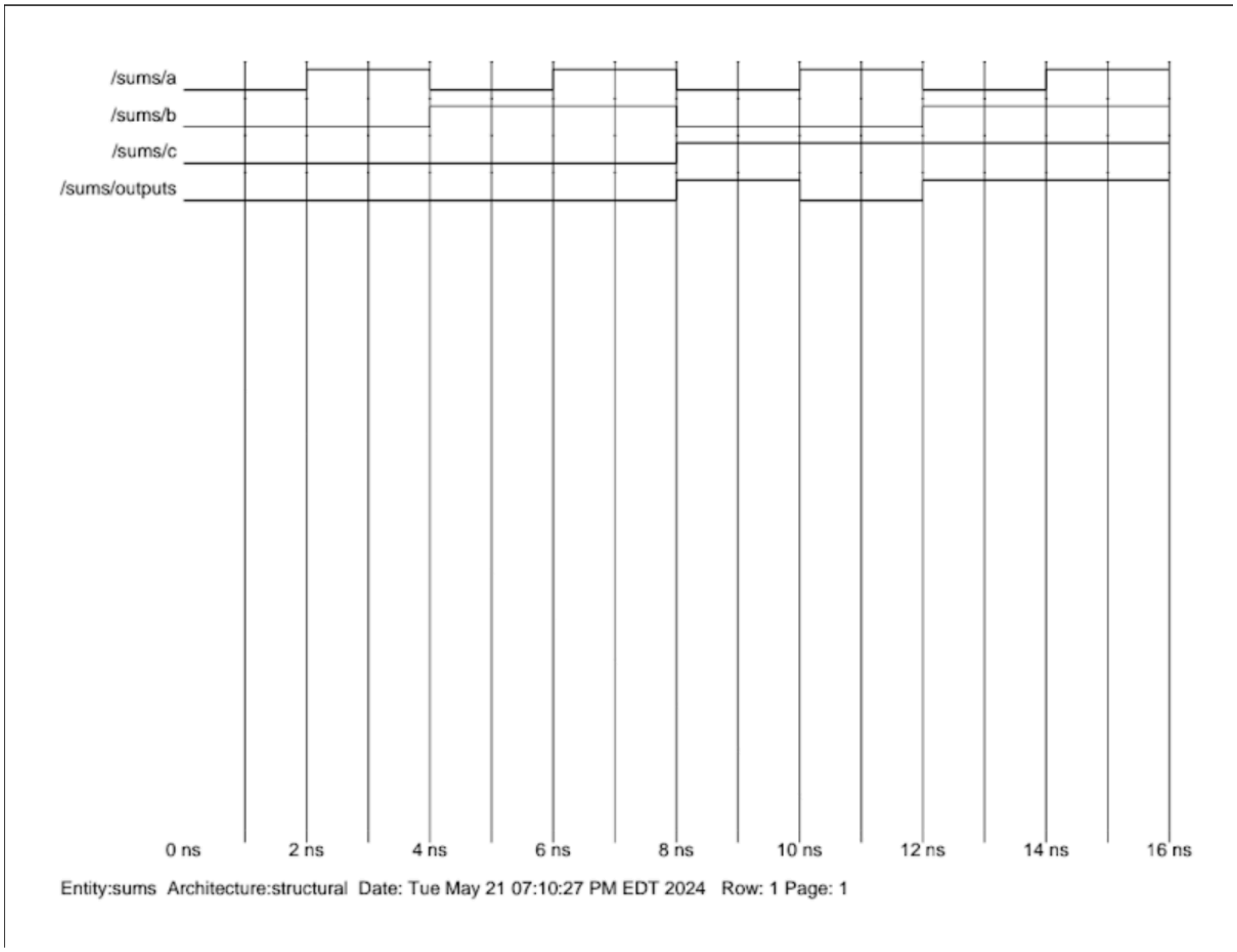
```
library IEEE;
use IEEE.std_logic_1164.all;
entity keith is

8

port( input_1 : in std_logic;
output : out std_logic);
end keith;
architecture rtl of keith is
signal first , second : std_logic;
begin
first <= not input_1 ;
second <= not first ;
output <= second;
end rtl ;

Answer ->
We can arrive at the conclusion that output = input_1, and we can see that
Vivado does the simplification by directly connecting them together.
```

CONCLUSION

We successfully wrote the and3, or3 and the sum_of_minterms vhdl files, compiled them and programmed the board, to then demonstrate the result to the TA. Furthermore we answer all the questions.

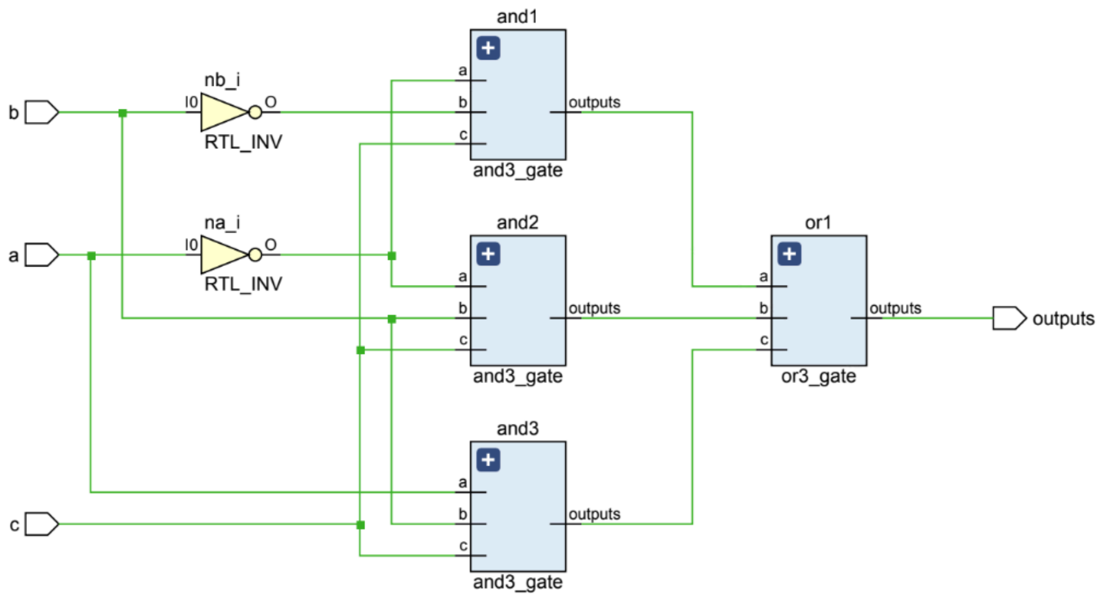
Deliverables

1. Modelsim simulation results for the port map version of the design.



2. RTL Elaborated and Implemented schematic diagrams as produced by the Vivado tool.

(PDFs of the schematics may be obtained by selecting right clicking the background within a schematic and selecting the “Save as PDF” choice.)



I forgot to export the RTL :(, sorry

3. VHDL code for both versions of the design.

With port :

```
library IEEE; use IEEE.std_logic_1164.all;

entity and3_gate  is
port( a, b,c: in std_logic;
      outputs : out std_logic);
end;


architecture example of and3_gate is begin
  outputs <= (a and b) and c;
end;
```

```
library IEEE; use IEEE.std_logic_1164.all;

entity or3_gate  is
port( a, b,c: in std_logic;
      outputs : out std_logic);
end;


architecture example of or3_gate is begin
  outputs <= (a or b) or c;
end;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity sums is
  port(a, b, c : in std_logic;
       outputs : out std_logic);
end sums;

architecture structural of sums is

-- declare and and or here

component and3_gate
  port ( a, b, c:   in std_logic;
         outputs : out std_logic);
end component;

component or3_gate
  port ( a, b, c :   in std_logic;
         outputs : out std_logic);
end component;

-- declare internal signals used to "hook up" components
-- and to communicate to the display decoder process
-- signal a,b,c  : std_logic;
signal na ,nb,nc : std_logic;
signal sig1,sig2,sig3     : std_logic;


-- declare configuration specification
-- NOTE: we want to use the half adder with true outputs
-- not the inverted ones we synthesized earlier!!


begin

-- component instantiation
na <= not a;
nb <= not b;
nc <= not c;
and1: and3_gate port map(a => na, b => nb,c => c,
                          outputs => sig1);
and2: and3_gate port map(a => na, b => b,c => c,
                          outputs => sig2);
and3: and3_gate port map(a => a, b => b,c => c,
                          outputs => sig3);
or1:  or3_gate port map(a => sig1, b => sig2,c => sig3,
                          outputs => outputs);


-- Nexys boards has active high LED outputs
-- so no need to negate top level output ports

end structural;
```

WITHOUT port :

```
library ieee;
use ieee.std_logic_1164.all;
```

```vhdl
entity sums is
  port(a, b, c : in std_logic;
       outputs : out std_logic);
end sums;




architecture structural of sums is


signal na ,nb,nc : std_logic;
signal sig1,sig2,sig3     : std_logic;

begin

-- component instantiation
na <= not a;
nb <= not b;
nc <= not c;
sig1 <= (na and nb) and c;
sig2 <= (na and b) and c;
sig3 <= (a and b) and c;
outputs <= (sig1 or sig2) or sig3;

end structural;
```