

DIGITAL ACCESS CONTROL

COURSE: CSE 215



NAME: KARIM WALID ELHAMMADY

ID: 16P3090

PROGRAM: CESS

EMAIL: KARIM.ELHAMMADY629@GMAIL.COM

1. **Introduction**

The purpose of this document is to explain the steps followed to perform logic synthesis of the RTL design implemented in phase 1; it was made using alliance tools ; syf, boom, boog, loon, xsch, flatbeh, proof and scapin, and Symphony for Simulation.

1. **XSCH\_LOON:**

**A screenshot of a computer screen

Description automatically generated**

1. **XSCH\_SCAPIN:**

A screenshot of a computer screen

Description automatically generated

1. **Delay and Area (boog):**

|  |  |  |
| --- | --- | --- |
| **enc** | **Area**  **( lamda²)** | **delay**  **(ps)** |
| a | 1396 | 57750 |
| j | 1597 | 53750 |
| m | 1729 | 56750 |
| o | 2171 | 88250 |
| r | 1932 | 64750 |

1. **Delay and Area (loon):**

|  |  |  |
| --- | --- | --- |
| **enc** | **area (lamda²)** | **delay (ps)** |
| a | 2148 | 58750 |
| j | 1832 | 53750 |
| m | 2485 | 59000 |
| o | 2253 | 91250 |
| r | 2335 | 64750 |

1. **Literal:**

|  |  |
| --- | --- |
| **enc** | **literal no** |
| a | 63 |
| j | 56 |
| m | 59 |
| o | 82 |
| r | 68 |

J encoding was chosen as it has the smallest area and delay time of all encodings.

1. **Simulation**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Make file:**

#-------Sdet--------------------------------------------#

all: sdeta.vbe \

sdetj.vbe \

sdetm.vbe \

sdeto.vbe \

sdetr.vbe

     @echo "<-- Generated"

#-------Finite State Machine Synthesis-----------------#

vhd\_to\_fsm:

    rename .vhd .fsm \*.vhd

sdeta.vbe: sdet.fsm

    @echo " Encoding Synthesis -> sdeta.vbe"

    syf -CEV -a sdet

sdetj.vbe: sdet.fsm

    @echo " Encoding Synthesis -> sdetj.vbe"

    syf -CEV -j sdet

sdetm.vbe: sdet.fsm

    @echo " Encoding Synthesis -> sdetm.vbe"

    syf -CEV -m sdet

sdeto.vbe: sdet.fsm

    @echo " Encoding Synthesis -> sdeto.vbe"

    syf -CEV -o sdet

sdetr.vbe: sdet.fsm

    @echo " Encoding Synthesis -> sdetr.vbe"

    syf -CEV -r sdet

#-------Clean Up---------------------------------------#

clean :

    rm -f \*.vbe \*.enc \*~

    @echo "Erase all the files generated by the makefile"

#-------BOOM-------------------------------------------#

sdet\_boom: sdeta\_b.vbe sdetj\_b.vbe sdetm\_b.vbe sdeto\_b.vbe sdetr\_b.vbe

%\_b.vbe: %.vbe

    @echo " Boolean Optimization -> $@"

    boom -V -d 50 $\* $\*\_b > $\*\_boom.out

#-------BOOG-------------------------------------------#

sdet\_boog: sdeta\_b.vst sdetj\_b.vst sdetm\_b.vst sdeto\_b.vst sdetr\_b.vst

%.vst: %.vbe paramfile.lax

    @echo " Logical Synthesis -> $@"

    boog -x 1 -l paramfile $\* > $\*\_boog.out

#-------LOON------------------------------------------#

sdet\_loon: sdeta\_b\_l.vst sdetj\_b\_l.vst sdetm\_b\_l.vst sdeto\_b\_l.vst sdetr\_b\_l.vst

%\_l.vst: %.vst paramfile.lax

    @echo " Netlist Optimization -> $@"

    loon -x 1 $\* $\*\_l paramfile > $\*\_loon.out

#-------Flatbeh&Proof---------------------------------#

%\_b\_l\_net.vbe: %\_b\_l.vst %.vbe

    @echo " Formal checking -> $@"

    flatbeh $\*\_b\_l $\*\_b\_l\_net > $\*\_flatbeh.out

    proof -d $\* $\*\_b\_l\_net > $\*\_proof.out

#-------scapin---------------------------------------#

ac\_scapin\_registers:

    cat sdetj\_b\_l.vst | grep sff

%\_scan.vst: %.vst scan.path

    @echo " scan-path insertion -> $@"

    scapin -VRB $\* scan $\*\_scan > scapin.out

1. **Parmfile:**

#M{2}

#L{2}

#C{

door:100;

alarm:100;

}

1. **Pathfile:**

BEGIN\_PATH\_REG

fsm\_alarm\_cs\_0\_ins

fsm\_alarm\_cs\_1\_ins

fsm\_alarm\_cs\_2\_ins

END\_PATH\_REG

BEGIN\_CONNECTOR

SCAN\_IN scanin

SCAN\_OUT scanout

SCAN\_TEST test

END\_CONNECTOR

1. **Testbench:**

ENTITY testbench IS

END ENTITY testbench;

ARCHITECTURE tb OF testbench IS

COMPONENT FSM\_alarm IS port

(code: in bit\_vector(3 downto 0);

reset: in bit;

day\_time: in bit;

vdd: in bit;

vss: in bit;

clk: in bit;

door: out bit;

alarm: out bit);

end component;

COMPONENT FSM\_alarm1 IS port

(code: in bit\_vector(3 downto 0);

reset: in bit;

day\_time: in bit;

vdd: in bit;

vss: in bit;

clk: in bit;

door: out bit;

alarm: out bit);

end component;

COMPONENT FSM\_alarm2 IS port

( code : in bit\_vector(3 downto 0);

reset : in bit;

day\_time : in bit;

vdd : in bit;

vss : in bit;

clk : in bit;

door : out bit;

alarm : out bit;

scanin : in bit;

test : in bit;

scanout : out bit);

end component;

FOR test0: FSM\_alarm USE ENTITY work.FSM\_alarm (behavioral\_description);

FOR test1: FSM\_alarm1 USE ENTITY work.sdetj\_b\_l (structural);

FOR test2: FSM\_alarm2 USE ENTITY work.sdetj\_b\_l\_scan (structural);

SIGNAL code : bit\_vector (3 downto 0) := "0000";

SIGNAL reset : bit := '0';

SIGNAL day\_time : bit := '0';

SIGNAL vdd : bit := '1';

SIGNAL vss : bit := '0';

SIGNAL clk : bit := '0';

SIGNAL scanin : bit ;

SIGNAL test : bit := '0';

SIGNAL scanout : bit ;

SIGNAL door : bit ;

SIGNAL alarm : bit ;

SIGNAL door1 : bit ;

SIGNAL alarm1 : bit ;

SIGNAL door2 : bit ;

SIGNAL alarm2 : bit ;

constant sequence : bit\_vector := "10101001010101100101111101001010101010101";

begin

test0: FSM\_alarm PORT MAP (code,reset,day\_time,vdd,vss,clk,door,alarm);

test1: FSM\_alarm1 PORT MAP (code,reset,day\_time,vdd,vss,clk,door1,alarm1);

test2: FSM\_alarm2 PORT MAP (code,reset,day\_time,vdd,vss,clk,door2,alarm2,scanin,test,scanout);

clk\_gen: process is begin

wait for 50 ns;

if clk = '0' then

clk <= '1';

else

clk <= '0';

end if;

end process;

testing: process is begin

reset <= '1';

wait for 100 ns;

reset <= '0';

code <="0010";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="0110";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="1010";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="0000";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="0101";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

wait for 100 ns;

--test set 1

code <="1101";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error ;

wait for 100 ns;

code <="1101";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

wait for 100 ns ;

--test set 2

code <="0010";

day\_time<='0';

wait for 100 ns;

code <="0110";

day\_time<='0';

wait for 100 ns;

code <="1101";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

wait for 100 ns;

--test set 3

code <="0010";

day\_time<='0';

wait for 100 ns;

code <="1101";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

wait for 100 ns;

--test set 4

code <="0010";

day\_time<='0';

wait for 100 ns;

code <="0110";

day\_time<='0';

wait for 100 ns;

code <="1010";

day\_time<='0';

wait for 100 ns;

code <="0001";

day\_time<='0';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

wait for 100 ns;

--test set 5

code <="0010";

day\_time<='0';

wait for 100 ns;

code <="0110";

day\_time<='0';

wait for 100 ns;

code <="1010";

day\_time<='0';

wait for 100 ns;

code <="0000";

day\_time<='0';

wait for 100 ns;

code <="0001";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

wait for 100 ns;

--test set 6

code <="0010";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="0110";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="1010";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="0000";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

code <="0101";

day\_time<='1';

wait for 100 ns;

assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2

report "wrong answer"

severity error;

wait for 100 ns;

--test case 7

test <= '1' ;

for i in 0 to sequence'length-1 loop

scanin <= sequence(i);

wait for 100 ns;

if i>=2 then

assert scanout = sequence (i-2)

report "scanout does not follow scan in"

severity error ;

end if;

end loop;

wait;

end process;

end architecture tb;

1. **Flatbeh:**

@@@@@@@@@ @@@@ @@@@@@@ @@@

@@ @ @@ @ @@ @@ @@

@@ @ @@ @@ @@ @@ @@

@@ @@ @@@@ @@ @@ @@ @@@@@ @@ @@@

@@ @ @@ @@ @ @@@@@@@@ @@ @@ @ @ @@@ @@

@@@@@@ @@ @@ @@ @@ @@@@@@ @@ @@ @@ @@

@@ @ @@ @@@@@ @@ @@ @@ @@@@@@@@@ @@ @@

@@ @@ @@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @@ @@ @@ @@ @@ @ @@ @@

@@ @@ @@ @@@ @@ @ @@ @@ @@ @@ @@ @@

@@@@@@ @@@@@@ @@@@ @@ @@@@ @@@@@@@@ @@@@ @@@@ @@@@

a netlist abstractor

Alliance CAD System 5.0 20090901, flatbeh 5.0 [2000/11/01]

Copyright (c) 1993-2019, ASIM/LIP6/UPMC

Author(s): Fran�ois DONNET, Huu Nghia VUONG

E-mail : alliance-users@asim.lip6.fr

========================= Environnement =========================

MBK\_WORK\_LIB = .

MBK\_CATA\_LIB = .:/usr/lib64/alliance/cells/sxlib:/usr/lib64/alliance/cells/dp\_sxlib:/usr/lib64/alliance/cells/rflib:/usr/lib64/alliance/cells/rf2lib:/usr/lib64/alliance/cells/ramlib:/usr/lib64/alliance/cells/romlib:/usr/lib64/alliance/cells/pxlib:/usr/lib64/alliance/cells/padlib

MBK\_CATAL\_NAME = CATAL

============================== Files ==============================

Netlist file = sdetj\_b\_l.vst

Output file = sdetj\_b\_l\_net.vbe

===================================================================

Loading './sdetj\_b\_l.vst'

flattening figure sdetj\_b\_l

loading inv\_x4

loading inv\_x2

loading an12\_x1

loading a3\_x2

loading na2\_x1

loading o2\_x2

loading o3\_x2

loading no4\_x1

loading na3\_x1

loading a2\_x2

loading nao22\_x1

loading sff1\_x4

loading no2\_x1

Restoring array's orders

BEH : Saving 'sdetj\_b\_l\_net' in a vhdl file (vbe)

1. **Proof:**

@@@@@@@ @@@

@@ @@ @ @@

@@ @@ @@ @@

@@ @@ @@@ @@@ @@@ @@@ @@

@@ @@ @@@ @@ @@ @@ @@ @@ @@@@@@@@

@@@@@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @@ @@ @@ @@

@@@@@@ @@@@ @@@ @@@ @@@@@@

Formal Proof

Alliance CAD System 5.0 20090901, proof 5.0

Copyright (c) 1990-2019, ASIM/LIP6/UPMC

E-mail : alliance-users@asim.lip6.fr

================================ Environment ================================

MBK\_WORK\_LIB = .

MBK\_CATA\_LIB = .:/usr/lib64/alliance/cells/sxlib:/usr/lib64/alliance/cells/dp\_sxlib:/usr/lib64/alliance/cells/rflib:/usr/lib64/alliance/cells/rf2lib:/usr/lib64/alliance/cells/ramlib:/usr/lib64/alliance/cells/romlib:/usr/lib64/alliance/cells/pxlib:/usr/lib64/alliance/cells/padlib

======================= Files, Options and Parameters =======================

First VHDL file = sdetj.vbe

Second VHDL file = sdetj\_b\_l\_net.vbe

The auxiliary signals are erased

Errors are displayed

===============================================================================

Compiling 'sdetj' ...

Compiling 'sdetj\_b\_l\_net' ...

---> final number of nodes = 295(151)

Running Abl2Bdd on `sdetj\_b\_l\_net`

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

Formal proof with Ordered Binary Decision Diagrams between

'./sdetj' and './sdetj\_b\_l\_net'

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

============================== PRIMARY OUTPUT ===============================

============================= AUXILIARY SIGNAL ==============================

============================== REGISTER SIGNAL ==============================

=============================== EXTERNAL BUS =================================

================================ INTERNAL BUS =================================

Formal Proof : OK

pppppppppppppppppppppppprrrrrrrrrrrrooooooooooooooooooooooooooooofffffffffffffff

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