Ain Shams University
Faculty of Engineering

جامعة عين شمس
كلية الهندسة

CSE 215

Electronic Design Automation

Project:

Digital Access Control

Name:

Abdelrahman Ibrahim ELGhamry

Email:

Ghamry98@hotmail.com

ID:

16P3043

Group:

1

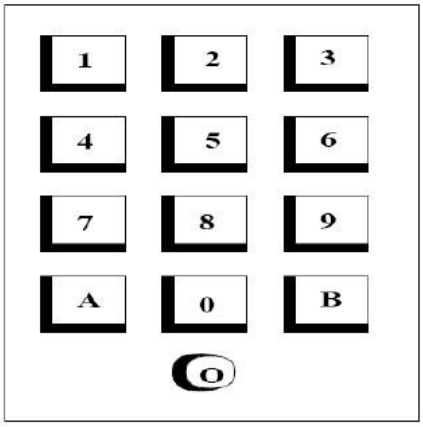
Department:

CESS

Table of Contents

1.0 INTRODUCTION

1.1 Brief Description



A Digital access control system (finite state machine) that takes a specific pin of five numbers as input and it has two operating modes:

1. **Daytime**: in the morning, door opens when pressing 'O' or entering the correct pin.
2. **Night**: opens only if the code is correct.

An alarm is triggered in case of:

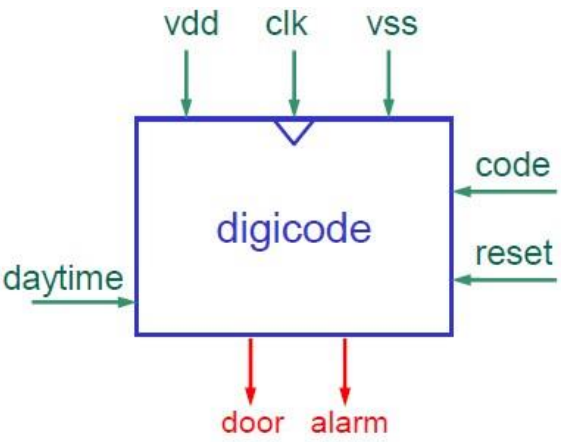
1. An incorrect entry, as soon as a wrong number is pressed.
2. If 'O' is pressed at night at any instance, even after any number of the correct pin before it is complete.

It contains a synchronous reset that deletes the entered code and restarts from the beginning waiting for the first digit and sets both door and alarm to 0, It is applied externally (reset=1) after:

1. Door opens (door=1).
2. Alarm is triggered (alarm=1).

The correct pin is: 26A05.

1.2 Circuit Inputs



The circuit has code, daytime, reset, clk, vdd and vss as inputs.

Daytime is applied externally during the morning (daytime=1).

Code can take values:

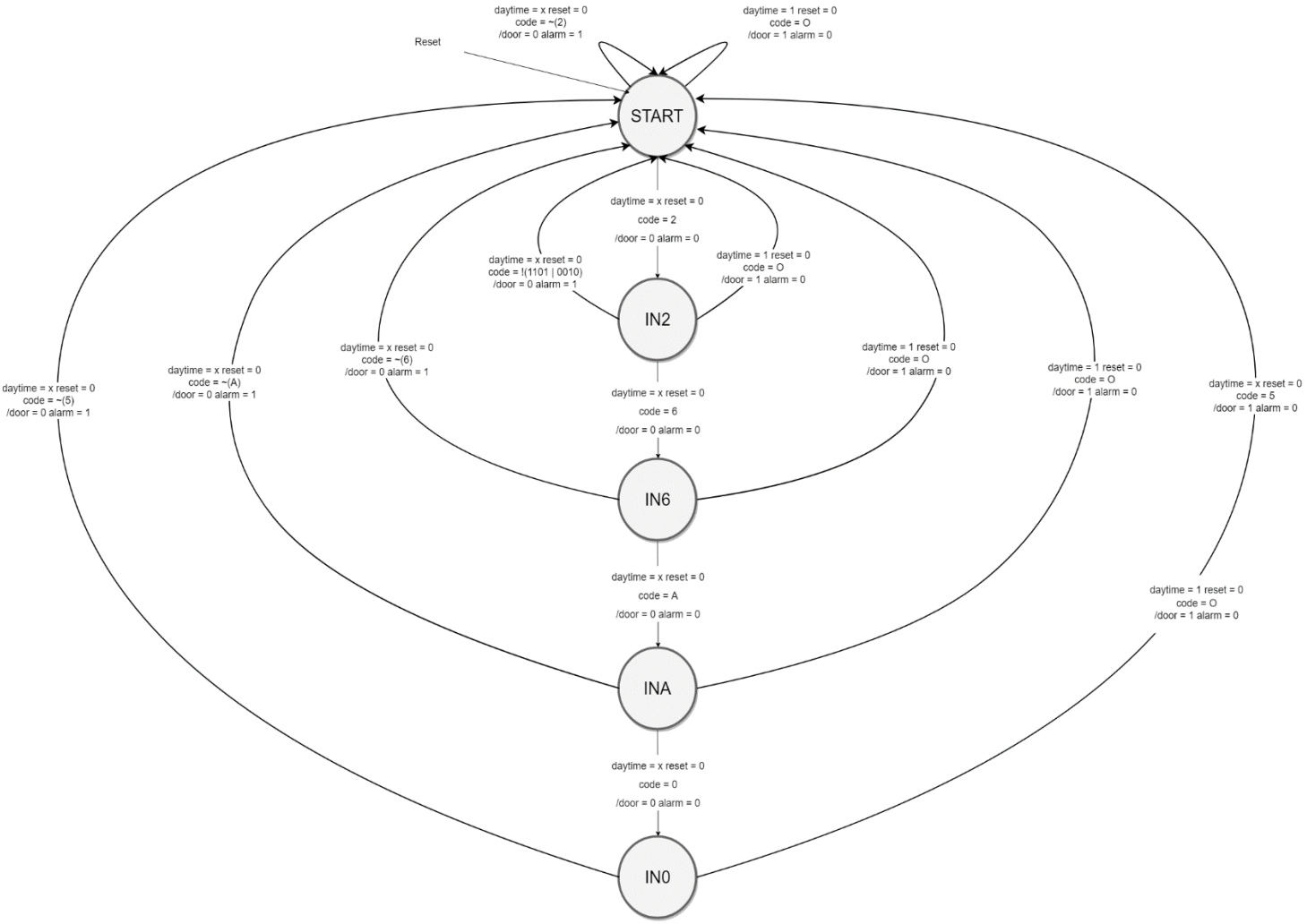
1. Numbers from 0 to 9 (binary coded).
2. A=1010, B=1011 and O=1101.

1.3 Output Type

As output is depending on the input and the current state, I decided to use Mealy output, this may be helpful to reduce the number of states.

The circuit has door and alarm as outputs.

2.0 FSM STATE DIAGRAM



3.0 VHDL SOURCE CODE

```
library IEEE;
library Sxlib_ModelSim;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.NUMERIC_STD.all;

ENTITY code IS
	PORT(
		daytime:   IN bit;
		code:		IN bit_vector(3 downto 0);
		reset:		IN bit;
		clk:		IN bit;
		vdd:		IN bit;
		vss:		IN bit;
		door:		OUT bit;
		alarm:		OUT bit
	);
END code;

ARCHITECTURE behav OF code IS
	TYPE State IS (START, IN2, IN6, INA, IN0);
	SIGNAL CS, NS: State;

	BEGIN
-- Process (1): State update (sequential)
		PROCESS (clk)
		 BEGIN
			if(clk = '1' and not clk'stable) THEN
			 CS <= NS;
			END if;
		END PROCESS;

-- Process (2): Transition and Generation functions
		PROCESS (CS, reset, code, daytime)
			BEGIN
				if(reset = '1') THEN
					NS <= START;
				else
					case CS IS
	WHEN START =>
		if(daytime = '1' and code = X"D") THEN
		NS <= START;
		door <= '1';
		alarm <= '0';
	elsif (code = X"2") THEN
		NS <= IN2;
		door <= '0';
		alarm <= '0';
	else
		NS <= START;
		door <= '0';
		alarm <= '1';
	END if;

	WHEN IN2 =>
		if(daytime = '1' and code = X"D") THEN
			NS <= START;
			door <= '1';
			alarm <= '0';
		elsif (code = X"6") THEN
			NS <= IN6;
			door <= '0';
			alarm <= '0';
		else
			NS <= START;
			door <= '0';
		alarm <= '1';
	END if;

	WHEN IN6 =>
		if(daytime = '1' and code = X"D") THEN
			NS <= START;
			door <= '1';
			alarm <= '0';
		elsif (code = X"A") THEN
			NS <= INA;
			door <= '0';
			alarm <= '0';
		else
			NS <= START;
			door <= '0';
			alarm <= '1';
		END if;

	WHEN INA =>
		if(daytime = '1' and code = X"D") THEN
			NS <= START;
			door <= '1';
			alarm <= '0';
		elsif (code = X"0") THEN
			NS <= IN0;
			door <= '0';
			alarm <= '0';
		else
			NS <= START;
			door <= '0';
			alarm <= '1';
		END if;

	WHEN IN0 =>
		if(daytime = '1' and code = X"D") THEN
			NS <= START;
			door <= '1';
			alarm <= '0';
		elsif (code = X"5") THEN
			NS <= START;
			door <= '1';
			alarm <= '0';
		else
			NS <= START;
			door <= '0';
			alarm <= '1';
		END if;
	END case;
END if; END PROCESS; END behav;
```

4.0 TEST STRATEGY

4.1 Test Table

Test Case #	Current State	Daytime	Code (HexaDecimal)	Reset	Next State	Door	Alarm
1	START IN2 IN6 INA IN0	0 0 0 0 0	2 6 A 0 5	0 0 0 0 0	IN2 IN6 INA IN0 START	0 0 0 0 1	0 0 0 0 0
2	START IN2 IN6 INA	1 1 1 1	2 6 A 3	0 0 0 0	IN2 IN6 INA START	0 0 0 0	0 0 0 1
3	START IN2 IN6 INA IN0	1 1 1 1 1	2 6 A 0 D “O”	0 0 0 0 0	IN2 IN6 INA IN0 START	0 0 0 0 1	0 0 0 0 0
4	START IN2 IN6 INA IN0	0 0 0 0 0	2 6 A 0 D “O”	0 0 0 0 0	IN2 IN6 INA IN0 START	0 0 0 0 0	0 0 0 0 1
5	START	0	3	0	START	0	1
6	START IN2	1 1	2 6	0 1	IN2 START	0 0	0 0
7	START	1	D “O”	0	START	1	0

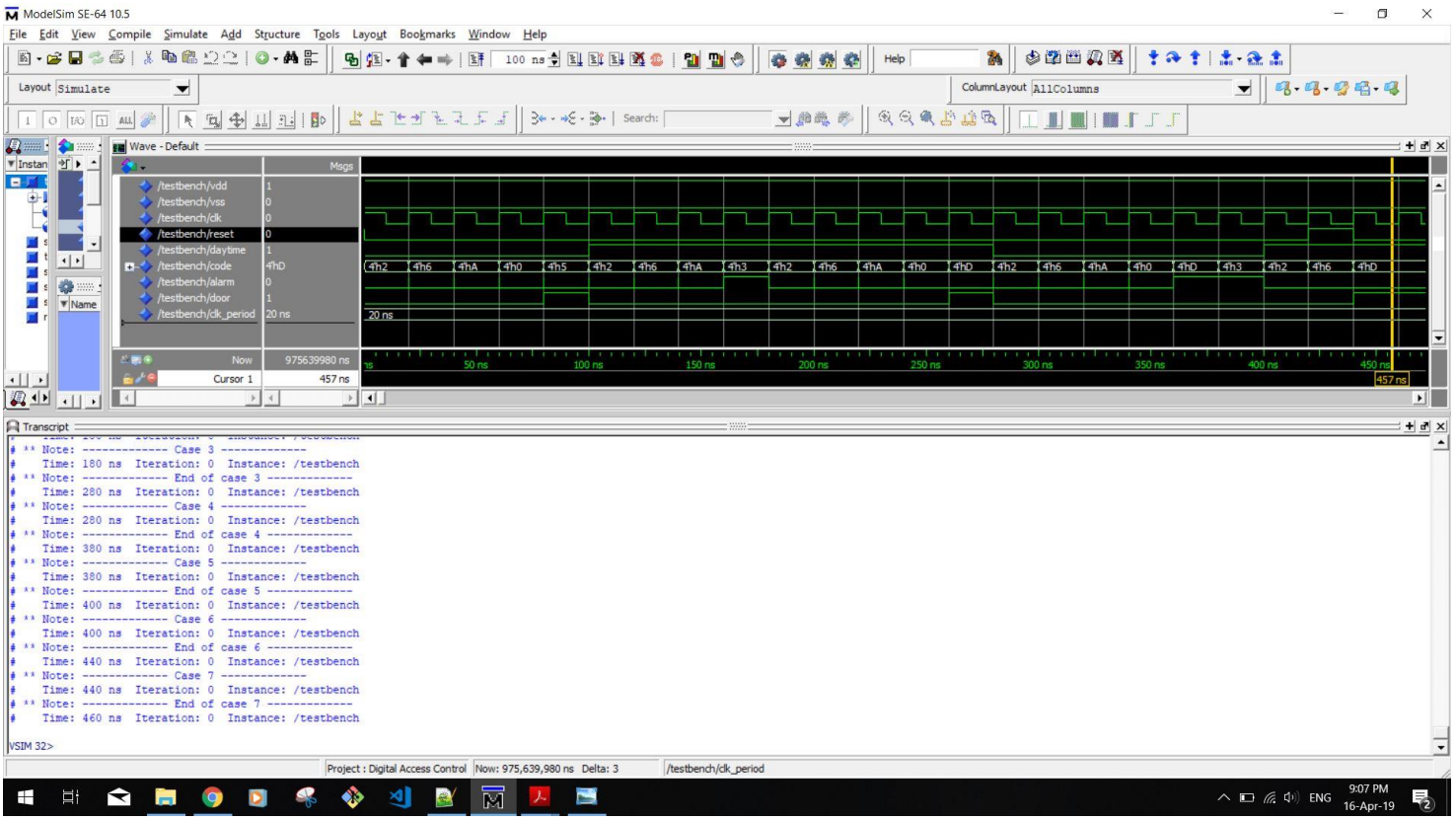
4.2 Testbench Code

```
library IEEE;
library Sxlib_ModelSim;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.NUMERIC_STD.all;

ENTITY testbench IS
END ENTITY testbench;

ARCHITECTURE testbench_beh OF testbench IS

-- Component Declaration for the Device Under Test (DUT)
	COMPONENT DAC IS
-- Just copy and paste the input and output ports of your module as such.
		PORT(
		daytime:	IN bit;
		code:		IN bit_vector(3 downto 0);
		reset:		IN bit;
		clk:		IN bit;
		vdd:		IN bit;
		vss:		IN bit;
		door:		OUT bit;
		alarm:		OUT bit
		);
	END COMPONENT DAC;

	FOR dut: DAC USE ENTITY WORK.code (behav);

-- Declare input signals and initialize them
	SIGNAL vdd		: bit := '1';
	SIGNAL vss		: bit := '0';
	SIGNAL clk		: bit := '1';
	SIGNAL reset		: bit := '1';
	SIGNAL daytime		: bit := '0';
	SIGNAL code		: bit_vector (3 downto 0) := X"0";

-- Declare output signals and initialize them
	SIGNAL alarm		: bit := '0';
	SIGNAL door		: bit := '0';

-- Constants and Clock period definitions
	CONSTANT clk_period : time := 20 ns;

	BEGIN

-- Instantiate the Device Under Test (DUT)
dut: DAC PORT MAP (daytime, code, reset, clk, vdd, vss, door, alarm);

-- Clock process definition( clock with 50% duty cycle )
		PROCESS
			BEGIN
				clk <= '1';
				WAIT FOR clk_period/2;
				clk <= '0';
				WAIT FOR clk_period/2;
		END PROCESS;

		PROCESS IS
			BEGIN
```

```
REPORT "------------ Case 1 ------------";
		daytime <= '0'; code <= X"2"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '0'
		REPORT "Error in Case 1.1"
		SEVERITY Error;

		daytime <= '0'; code <= X"6"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '0'
		REPORT "Error in Case 1.2"
		SEVERITY Error;

		daytime <= '0'; code <= X"A"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '0'
		REPORT "Error in Case 1.3"
		SEVERITY Error;

		daytime <= '0'; code <= X"0"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '0'
		REPORT "Error in Case 1.4"
		SEVERITY Error;

		daytime <= '0'; code <= X"5"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '1' and alarm = '0'
		REPORT "Error in Case 1.5"
		SEVERITY Error;
REPORT "------------ End of case 1 ------------";

REPORT "------------ Case 2 ------------";
		daytime <= '1'; code <= X"2"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '0'
		REPORT "Error in Case 2.1"
		SEVERITY Error;

		daytime <= '1'; code <= X"6"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '0'
		REPORT "Error in Case 2.2"
		SEVERITY Error;

		daytime <= '1'; code <= X"A"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '0'
		REPORT "Error in Case 2.3"
		SEVERITY Error;

		daytime <= '1'; code <= X"3"; reset <= '0';
		WAIT FOR clk_period;
		ASSERT door = '0' and alarm = '1'
		REPORT "Error in Case 2.4"
		SEVERITY Error;
REPORT "------------ End of case 2 ------------";
```

```
REPORT "------------ Case 3 ------------";
        daytime <= '1'; code <= X"2"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 3.1"
        SEVERITY Error;

        daytime <= '1'; code <= X"6"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 3.2"
        SEVERITY Error;

        daytime <= '1'; code <= X"A"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 3.3"
        SEVERITY Error;

        daytime <= '1'; code <= X"0"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 3.4"
        SEVERITY Error;

        daytime <= '1'; code <= X"D"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '1' and alarm = '0'
        REPORT "Error in Case 3.5"
        SEVERITY Error;
REPORT "------------ End of case 3 ------------";

REPORT "------------ Case 4 ------------";
        daytime <= '0'; code <= X"2"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 4.1"
        SEVERITY Error;

        daytime <= '0'; code <= X"6"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 4.2"
        SEVERITY Error;

        daytime <= '0'; code <= X"A"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 4.3"
        SEVERITY Error;

        daytime <= '0'; code <= X"0"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 4.4"
        SEVERITY Error;

        daytime <= '0'; code <= X"D"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '1'
        REPORT "Error in Case 4.5"
        SEVERITY Error;
REPORT "------------ End of case 4 ------------";

REPORT "------------ Case 5 ------------";
        daytime <= '0'; code <= X"3"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '1'
        REPORT "Error in Case 5.1"
        SEVERITY Error;
REPORT "------------ End of case 5 ------------";

REPORT "------------ Case 6 ------------";
        daytime <= '1'; code <= X"2"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 6.1"
        SEVERITY Error;

        daytime <= '1'; code <= X"6"; reset <= '1';
        WAIT FOR clk_period;
        ASSERT door = '0' and alarm = '0'
        REPORT "Error in Case 6.2"
        SEVERITY Error;
REPORT "------------ End of case 6 ------------";

REPORT "------------ Case 7 ------------";
        daytime <= '1'; code <= X"D"; reset <= '0';
        WAIT FOR clk_period;
        ASSERT door = '1' and alarm = '0'
        REPORT "Error in Case 7.1"
        SEVERITY Error;
REPORT "------------ End of case 7 ------------";

        WAIT; -- don't repeat above test vectors
        END PROCESS;
END testbench_beh;
```

5.0 SIMULATION RESULTS



6.0 ALLIANCE TOOLS

SYF: Finite State Machine synthesizer.

BOOM: Boolean Minimization.

BOOG: Library Binding.

LOON: Local optimizations of Nets.

XSCH: Graphical netlist viewer.

FLATBEH: Behavioral from Structural.

PROOF: Formal Verification.

SCAPIN: Scan-path insertion (DFT).

OCP: Standard Cell Placer.

NERO: Over-Cell Router.

COUGAR: Symbolic Netlist Extractor.

LVX: Netlist comparator.

GRAAL: Symbolic layout editor.

DRUC: Symbolic Design-rule checker.

S2R: Symbolic-to-Real layout converter.

DREAL: Real layout editor.

7.0 LOGICAL CIRCUIT GENERATION

7.1 FSM Synthesis

Synthesized the FSM with SYF tool using different state encoding algorithms -a, -j, -m, -o, -r and by using the options -CEV in order to get different circuits with different areas and clock frequencies.

7.2 Boolean Network Optimization

Optimized the FSM with BOOM tool by giving an appropriate optimization in delay and area for each previously generated circuits.

Optimized the circuits using 50% for area optimization and 50% for delay optimization.

State/Encoding	-a	-j	-m	-o	-r
START	0	0	0	0 1	3
IN2	2	3	4	1 2	6
IN6	4	1	1	2 4	1
INA	6	2	2	3 8	0
IN0	1	4	5	4 10	7

7.3 Library Mapping

- Setting the Optimization Mode to 5.
- Setting the Optimization Level to 2.
- set the output load capacitance on the outputs (door and alarm) to 100 fF.

Launching BOOG on different netlists to observe SYF options influence (different state encoding techniques).

BOOG also translates Boolean functions associated to each node of a Boolean network into a network of gates.

7.4 Netlist Optimization

Launching LOON on different netlists, it operates only on critical paths to reduce the propagation time.

It uses two techniques:

1. Gate replacement with higher current versions to reduce delay.
2. Buffer insertion in case of very high fanouts.

Encoding	Delay (ps)	Area (lamda^2)	Delay*Area
a	170	62250	10582500
j	303	55750	16892250
m	170	62000	10540000
o	416	77250	32136000
r	303	69500	21058500

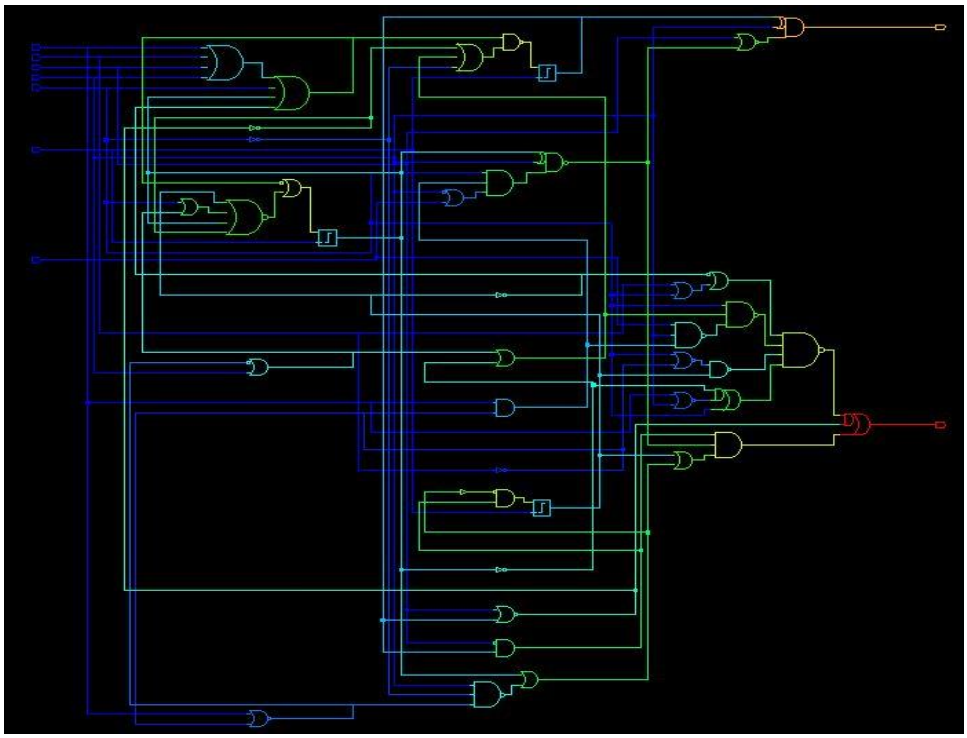
The best implementation goes to 'm' encoding as it has the lowest average (area*delay) factor relative to other encodings.

7.5 Netlist Visualization

Using the XSCH graphical schematic viewer tool to visualize the Gate level circuit of the m encoding.

The red-colored path, defines the critical path of the FSM.

Check the appendix for the circuit .vst file.



7.6 Netlist Checking

Making a formal comparison of the chosen netlist with the original behavioral file resulting to verify that it is working correctly and with the same behavior.

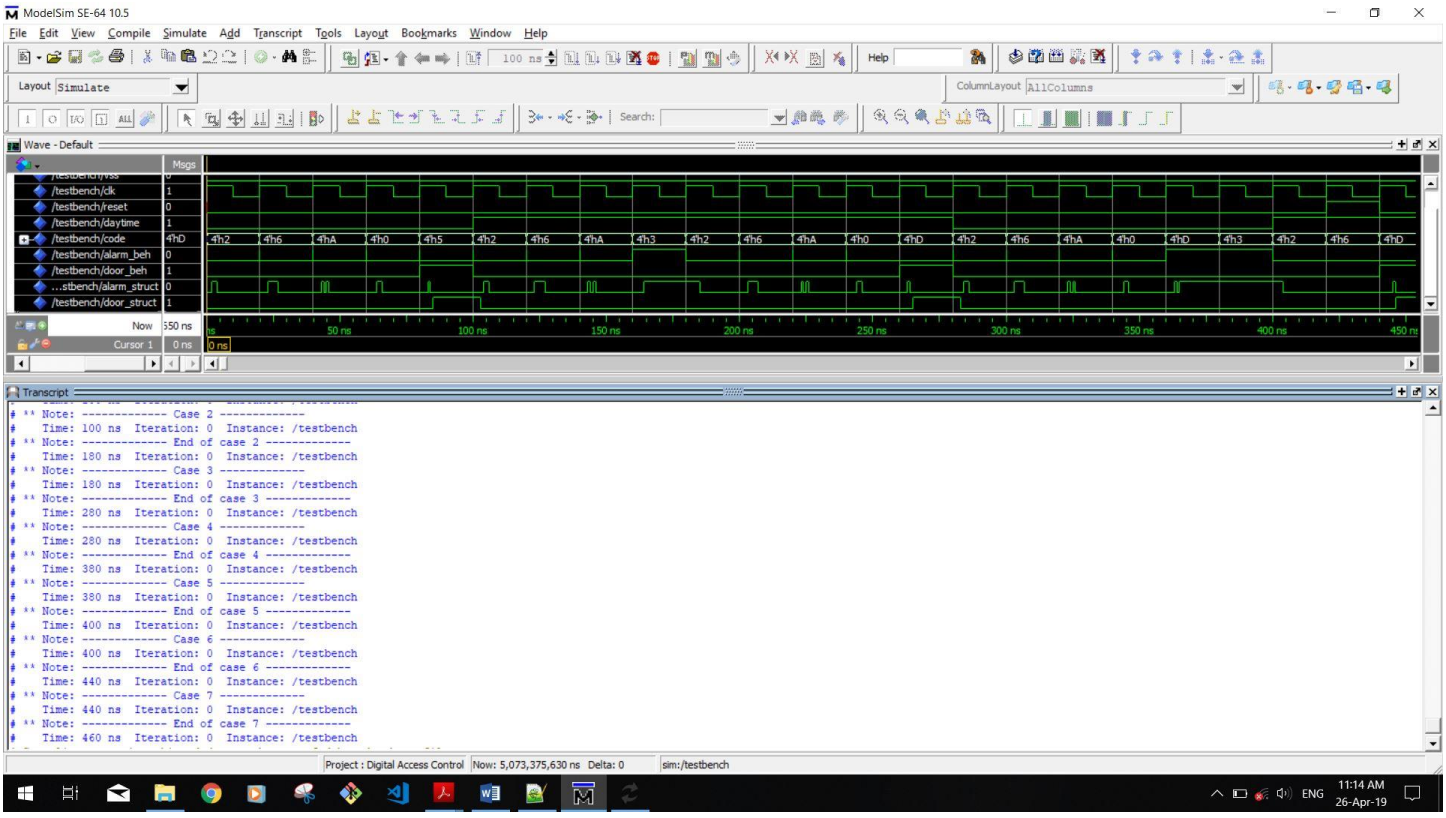
This could be done using the FLATBEH and PROOF tools.

Check the appendix for the proof output file.

7.7 Delay Simulation

The obtained gate-level netlist has been functionally verified. Standard-cell delay has been ignored during all the above steps. Note that both BOOG and LOON synthesis and optimization tools have estimated the critical path delay. It should be easy to verify that this delay is less than the required clock period. It is time now for a delay simulation to double-check the speed performance.

Check the appendix for the testbench file.

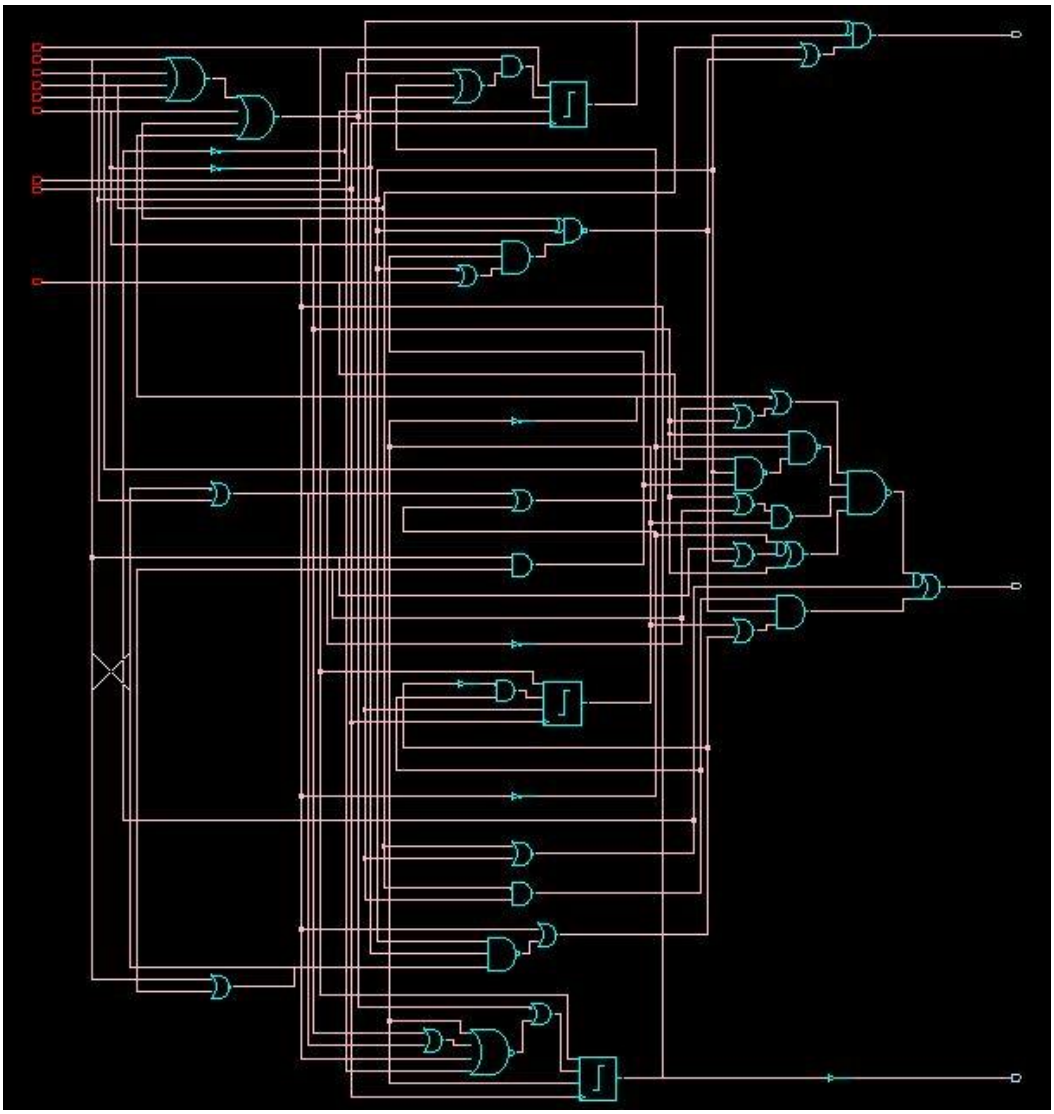


7.8 Scan-path Insertion (DFT)

With the SCAPIN tool, we can insert a scan-path into the netlist. The scan-path allows the designer to observe in test mode the stored values of all registers of the circuit. The path is created by changing each register into a mux_register (i.e. by inserting a multiplexer in front of all registers) and connecting them in series.

Check the appendix for the .path file.

Using the XSCH graphical schematic viewer tool to visualize the Gate level circuit of the chosen circuit after changing each register into a mux_register.

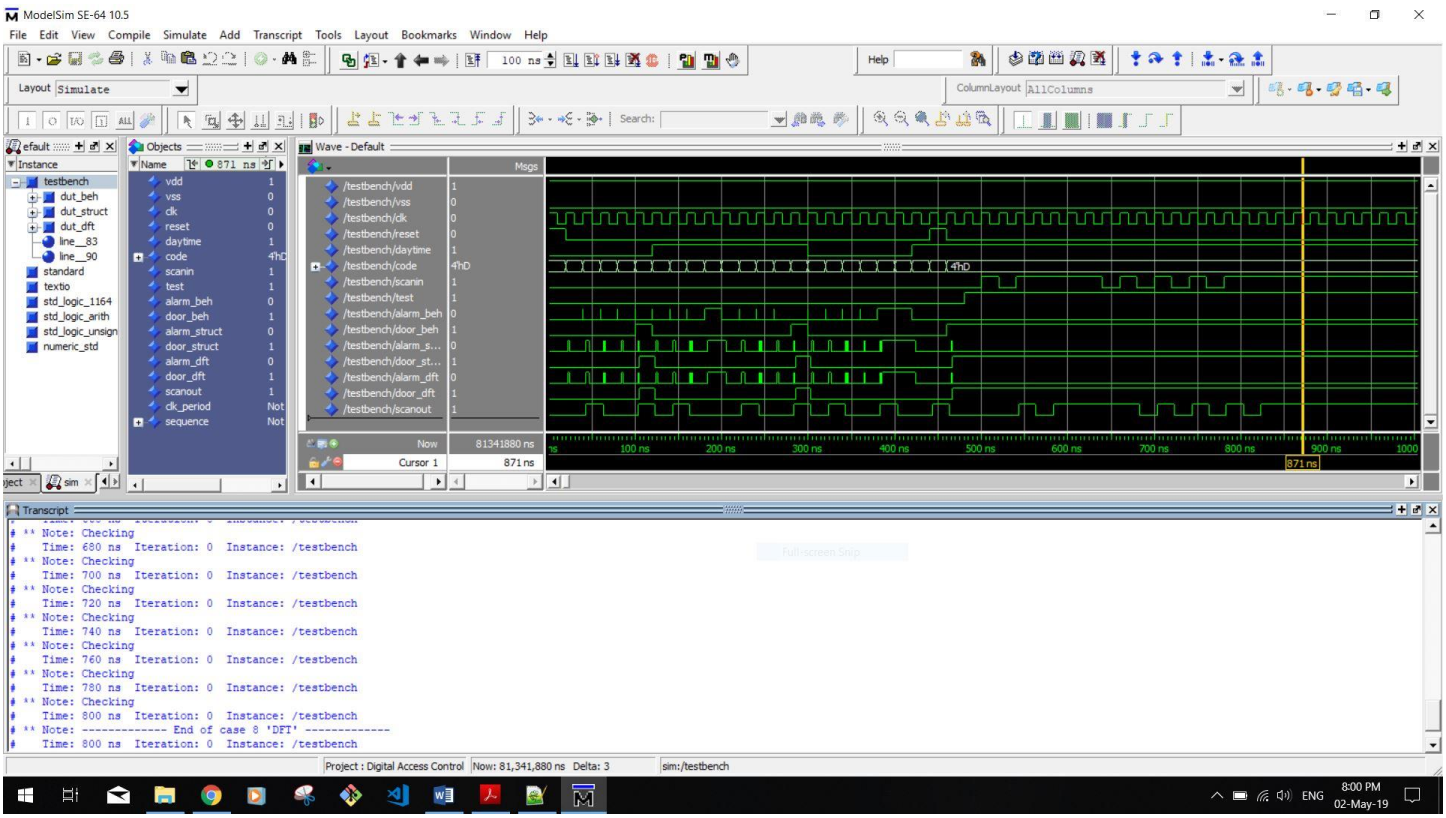


7.9 Scan-path Simulation

By using the same testbench as before, comparing behavioral and structural outputs.

In addition, we will:

1. Instantiate the new scan netlist, in addition to behavioral and structural descriptions.
2. Put test = 0, to verify normal operation with the same patterns as before.
3. Compare the three netlists outputs, they should be identical.
4. Put test = 1 and add an extra pattern “1010101011111010” for scanin.
5. Assert that the scanout signal will have the same output as the input pattern of scanin after 2 clock cycles.

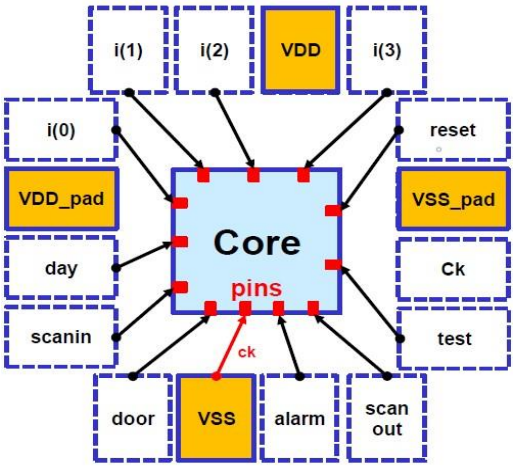


8.0 PHYSICAL CIRCUIT GENERATION

8.1 Floor Planning

Before starting physical synthesis, we must first construct the chip floor planning.

The opposite figure determines I/O pad distribution around the core.



8.2 Placement

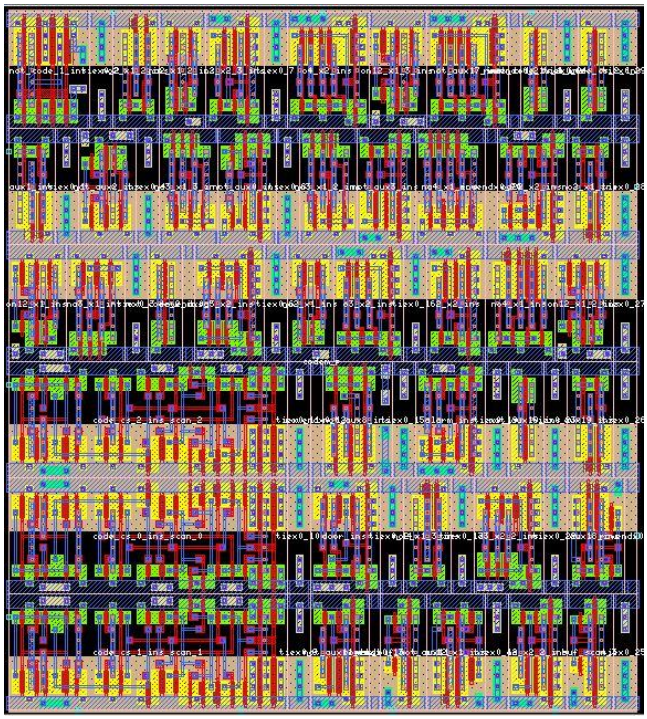
Using the OCP tool to start placement, giving it options:

-ioc: to specify a placement for external connectors (pins) described in a .ioc file.

-ring: to automatically place the connectors for the ring pad placement tool.

Using the GRAAL Symbolic layout editor tool to visualize the symbolic layout of the circuit.

Check the appendix for the .ioc file.

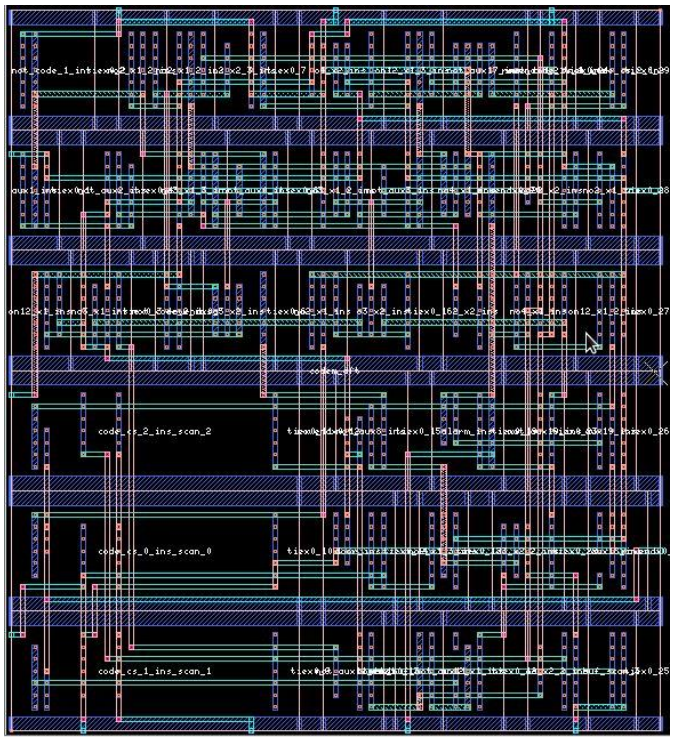


8.3 Routing

Using the NERO tool to start routing the placed cells.

The layout editor tool GRAAL can be then used to inspect the output layout.

It should be the same as the placement file with over-the-cell routing added.



8.4 Post-Layout Verification

Post-layout verification is composed of two steps: electrical and physical. Electrical verification checks that the extracted netlist is exactly the same as the gate-level netlist (LVS), while physical verification checks if there are design-rule errors in the generated layout (DRC).

8.4.1 Layout-vs-Schematics (LVS)

Netlist comparison is done on two steps:

1. Netlist extraction using the COUGAR tool.
2. Netlist comparison using LVX tool.

Check the appendix for the COUGAR and LVX output files.

8.4.2 Design-Rule Checking

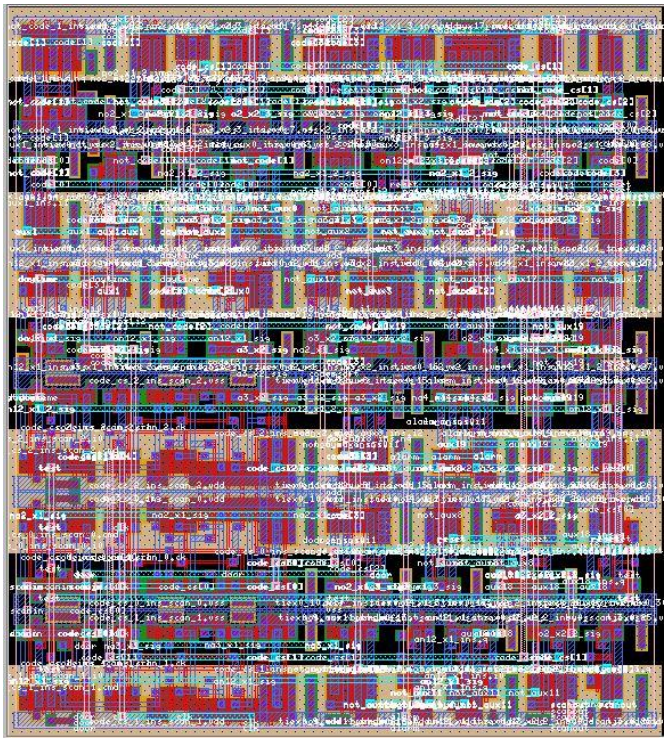
Using the DRUC design rule checker tool in order to check symbolic rules.

Check the appendix for the DRUC output file.

8.5 Symbolic-to-Real Conversion

Finally, using the S2R tool to generate the real fabrication technology from the previously generated symbolic layout.

Using the DREAL real layout viewer tool to visualize the real fabrication technology.



APPENDIX

1. The Behavioral-Structural testbench. Project\code_beh_struct_testbench.vhd
2. The Behavioral-Structural-DFT testbench. Project\code_beh_struct_dft_testbench.vhd
3. The Loon.vhd circuit. Project\codem_b_l.vhd
4. The dft circuit. Project\codem_dft.vhd
5. The output proof file of the proof command. Project\codem_proof.out
6. The path file. Project\path.path
7. The Make file. Project\Makefile
8. The Paramfile. Project\paramfile.lax
9. The core pin order .ioc file Project\pin.ioc
10. The cougar tool output file Project\cougar_codem_dft.out
11. The lvx tool output file Project\lvx_codem_dft.out
12. The druc tool output file Project\druc_core.out
13. The s2r tool output file Project\s2r.out