TECHNISCHE HOCHSCHULE DEGGENDORF THD

Faculty of Applied Computer Science

Course “FPGA-Programming” - Report

Summer Term 2020

Group 17

“Automation of a Gated Parking Lot”

Name of Authors:

Girish Patil Ganesh K Vernekar

Zineddine Bettouche Aaisha Ghodekar

Name of Examiner:

Prof. Dr. Martin Schramm

Deggendorf, July 27nt, 2020

Table of Contents

1. Introduction to Project Idea

The project consists of the interactive automated gated parking. Our project involves the concept of automatic car parking. A total of 9 parking spots are considered and it involves several sensors as input and a payment system incorporated. The 7-segment display is used to guide the user during the entire process. The user must enter a password and position of the address to gain entry into the parking lot. The memory is used to check if the parking place is available or not. The system can count the number of cars parked. The 7 Segment decoder can display communicates to the driver such as “Addr”, “Enter”, “Hold”, “Full”

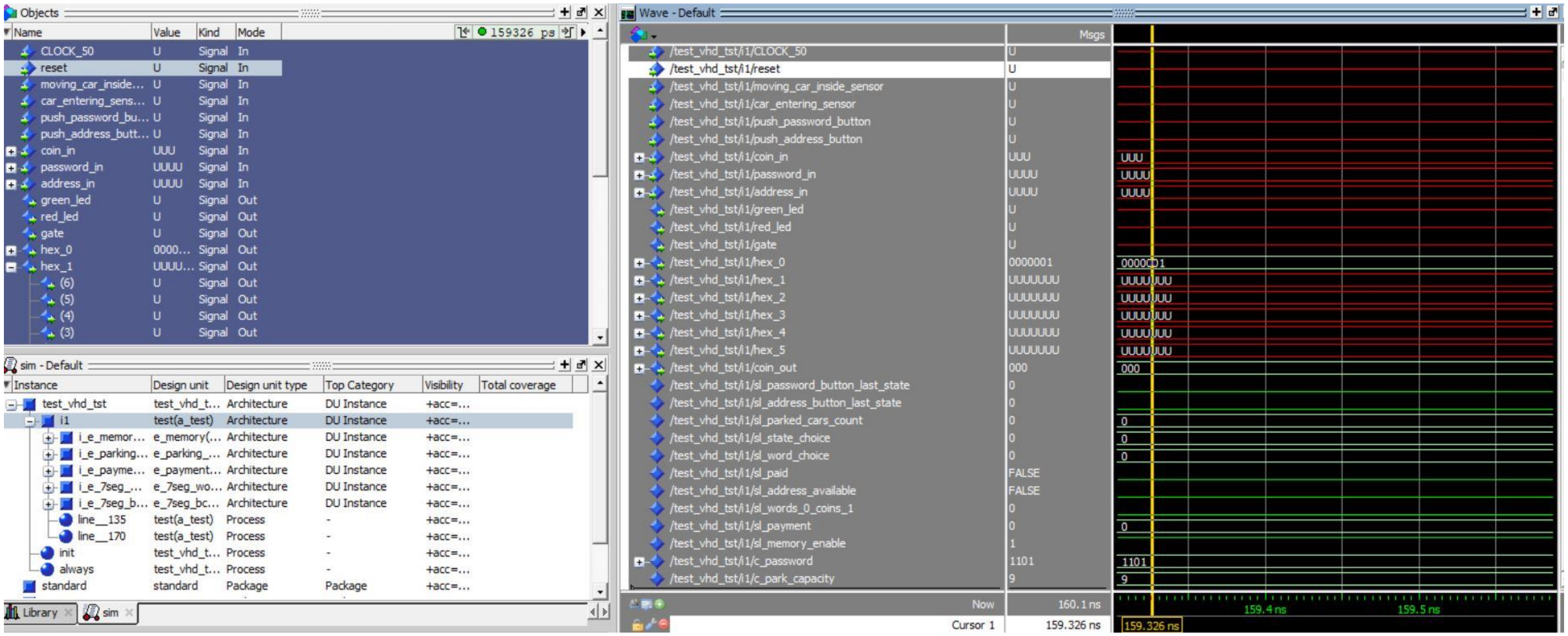
2. State Description

A total of 10 states are considered

- No_action_state: This state detects a car near the gate via a sensor (car_entering_sensor).
- Car_detected_state: The driver inputs a password using 4 switches and presses a button.
- Password_State: The password is verified in this state. If the entered password is correct, then it four 7segement displays used to display ‘Addr’. The driver is requested to enter the address of the parking place to take using 4 switches and presses a button.
- Address_state: In this state the system checks in the memory if the place is available to park. If available, then memory (address) is assigned to “1” meaning “now taken”

- Paying_state: The driver enters coins to pay for the parking place. The coins allowed to be entered are 10, 20 and 50 cents (std_logic_vector[2 downto 0]). Switching between possible cases of coin_in and setting coin_out to give back the change. A signal “paid” is set by default to false and is assigned to true when the driver pays the coins.
- Parking_car_state: Checks if there are moving cars inside the parking lot (moving_car_inside_sensor). If the car is moving, then it enters the Hold_state. If no car is moving, then four 7-segment displays “Enter”. The gate opens and the green LED turns on. The program waits for some time and then it toggles the gate and the green LED. The four 7-segments turns off.
- Hold_state: It checks if there are moving cars. If sensors detects moving cars then it remains in the same state. Else it transits to the parking_car_state.
- Parking_done_state: In this state it counts the number of parked cars and this is displayed on the 7-segment display. If the parking lot is full then it enters the full_parked_state. If it is not full then it transits to no_action_state.
- Full_parked_state: In this state four 7-segment displays “Full”.

3. Test Bench



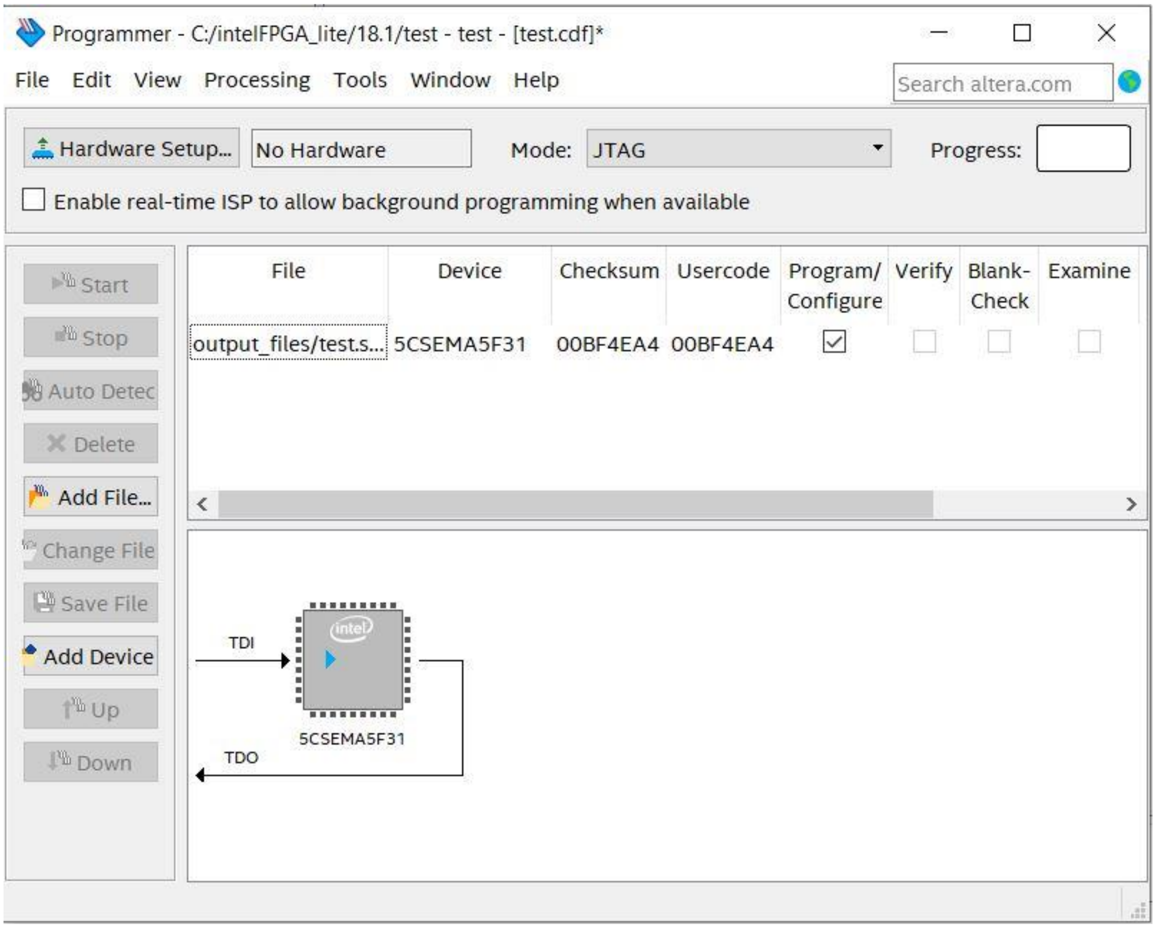
Name	Value	Kind	Mode
CLOCK_50	U	Signal	In
reset	U	Signal	In
moving_car_inside...	U	Signal	In
car_entering_sens...	U	Signal	In
push_password_bu...	U	Signal	In
push_address_butt...	U	Signal	In
coin_in	UUU	Signal	In
password_in	UUUU	Signal	In
address_in	UUUU	Signal	In
green_led	U	Signal	Out
red_led	U	Signal	Out
gate	U	Signal	Out
hex_0	0000...	Signal	Out
hex_1	UUUU...	Signal	Out
(6)	U	Signal	Out
(5)	U	Signal	Out
(4)	U	Signal	Out
(3)	U	Signal	Out

Instance	Design unit	Design unit type	Top Category	Visibility	Total coverage
test_vhd_tst	test_vhd_t...	Architecture	DU Instance	+acc=...	
i1	test(a_test)	Architecture	DU Instance	+acc=...	
i_e_memor...	e_memory(...	Architecture	DU Instance	+acc=...	
i_e_parking...	e_parking_...	Architecture	DU Instance	+acc=...	
i_e_payme...	e_payment...	Architecture	DU Instance	+acc=...	
i_e_7seg_...	e_7seg_wo...	Architecture	DU Instance	+acc=...	
i_e_7seg_b...	e_7seg_bc...	Architecture	DU Instance	+acc=...	
line__135	test(a_test)	Process	-	+acc=...	
line__170	test(a_test)	Process	-	+acc=...	
init	test_vhd_t...	Process	-	+acc=...	
always	test_vhd_t...	Process	-	+acc=...	
standard	standard	Package	Package	+acc=...	

	Msgs	
/test_vhd_tst/i1/CLOCK_50	U	
/test_vhd_tst/i1/reset	U	
/test_vhd_tst/i1/moving_car_inside_sensor	U	
/test_vhd_tst/i1/car_entering_sensor	U	
/test_vhd_tst/i1/push_password_button	U	
/test_vhd_tst/i1/push_address_button	U	
/test_vhd_tst/i1/coin_in	UUU	UUU
/test_vhd_tst/i1/password_in	UUUU	UUUU
/test_vhd_tst/i1/address_in	UUUU	UUUU
/test_vhd_tst/i1/green_led	U	
/test_vhd_tst/i1/red_led	U	
/test_vhd_tst/i1/gate	U	
/test_vhd_tst/i1/hex_0	0000001	0000001
/test_vhd_tst/i1/hex_1	UUUUUUU	UUUUUUU
/test_vhd_tst/i1/hex_2	UUUUUUU	UUUUUUU
/test_vhd_tst/i1/hex_3	UUUUUUU	UUUUUUU
/test_vhd_tst/i1/hex_4	UUUUUUU	UUUUUUU
/test_vhd_tst/i1/hex_5	UUUUUUU	UUUUUUU
/test_vhd_tst/i1/coin_out	000	000
/test_vhd_tst/i1/sl_password_button_last_state	0	
/test_vhd_tst/i1/sl_address_button_last_state	0	
/test_vhd_tst/i1/sl_parked_cars_count	0	0
/test_vhd_tst/i1/sl_state_choice	0	0
/test_vhd_tst/i1/sl_word_choice	0	0
/test_vhd_tst/i1/sl_paid	FALSE	
/test_vhd_tst/i1/sl_address_available	FALSE	
/test_vhd_tst/i1/sl_words_0_coins_1	0	
/test_vhd_tst/i1/sl_payment	0	0
/test_vhd_tst/i1/sl_memory_enable	1	
/test_vhd_tst/i1/c_password	1101	1101
/test_vhd_tst/i1/c_park_capacity	9	9
Now	160.1 ns	
Cursor 1	159.326 ns	159.326 ns

Figure 1 Test Bench

4. Signal Tap Logic Analyser



Programmer - C:/intelFPGA_lite/18.1/test - test - [test.cdf]*

No Hardware

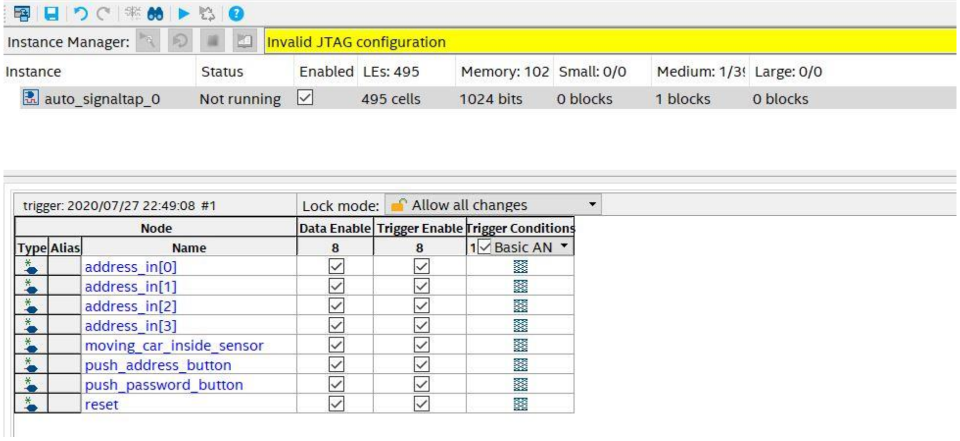
Mode: JTAG

Progress:

□ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine
output_files/test.s...	5CSEMA5F31	00BF4EA4	00BF4EA4	☑	□	□	□

Figure 2 Signal Tap Analyzer – Programmer



Instance Manager: Invalid JTAG configuration

Instance	Status	Enabled	LEs: 495	Memory: 102	Small: 0/0	Medium: 1/3!	Large: 0/0
auto_signaltap_0	Not running	☑	495 cells	1024 bits	0 blocks	1 blocks	0 blocks

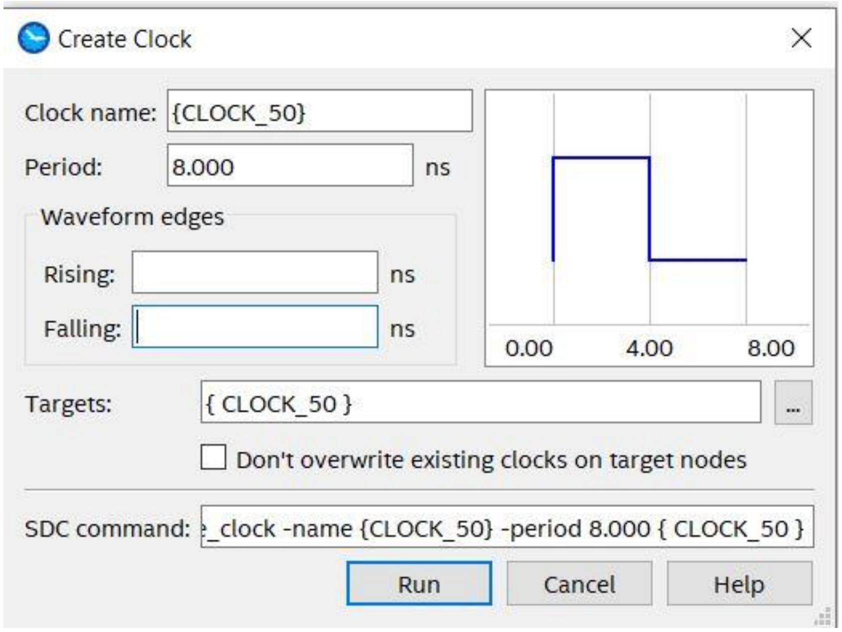
trigger: 2020/07/27 22:49:08 #1

Lock mode: Allow all changes

Node			Data Enable	Trigger Enable	Trigger Conditions
Type	Alias	Name	8	8	1☑ Basic AN
		address_in[0]	☑	☑	▨
		address_in[1]	☑	☑	▨
		address_in[2]	☑	☑	▨
		address_in[3]	☑	☑	▨
		moving_car_inside_sensor	☑	☑	▨
		push_address_button	☑	☑	▨
		push_password_button	☑	☑	▨
		reset	☑	☑	▨

Figure 3 Signal Configuration

5. TimeQuest Analyzer



Clock name: {CLOCK_50}

Period: 8.000 ns

Rising: ns

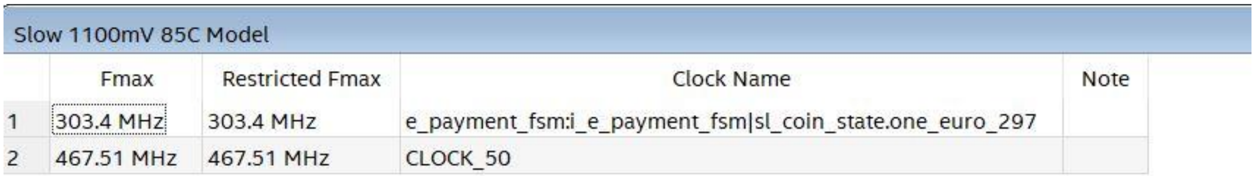
Falling: ns

Targets: { CLOCK_50 }

□ Don't overwrite existing clocks on target nodes

SDC command: _clock -name {CLOCK_50} -period 8.000 { CLOCK_50 }

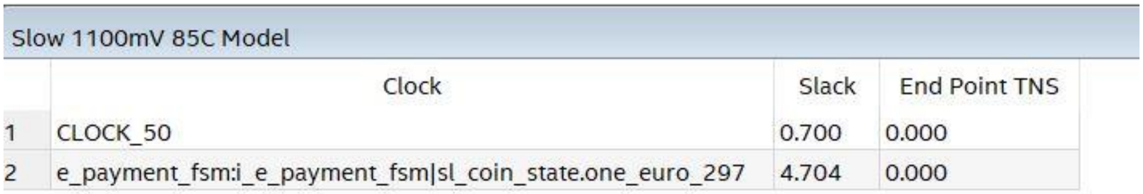
Figure 4 Editing Clock Constraint



Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	303.4 MHz	303.4 MHz	e_payment_fsm:i_e_payment_fsm|sl_coin_state.one_euro_297	
2	467.51 MHz	467.51 MHz	CLOCK_50	

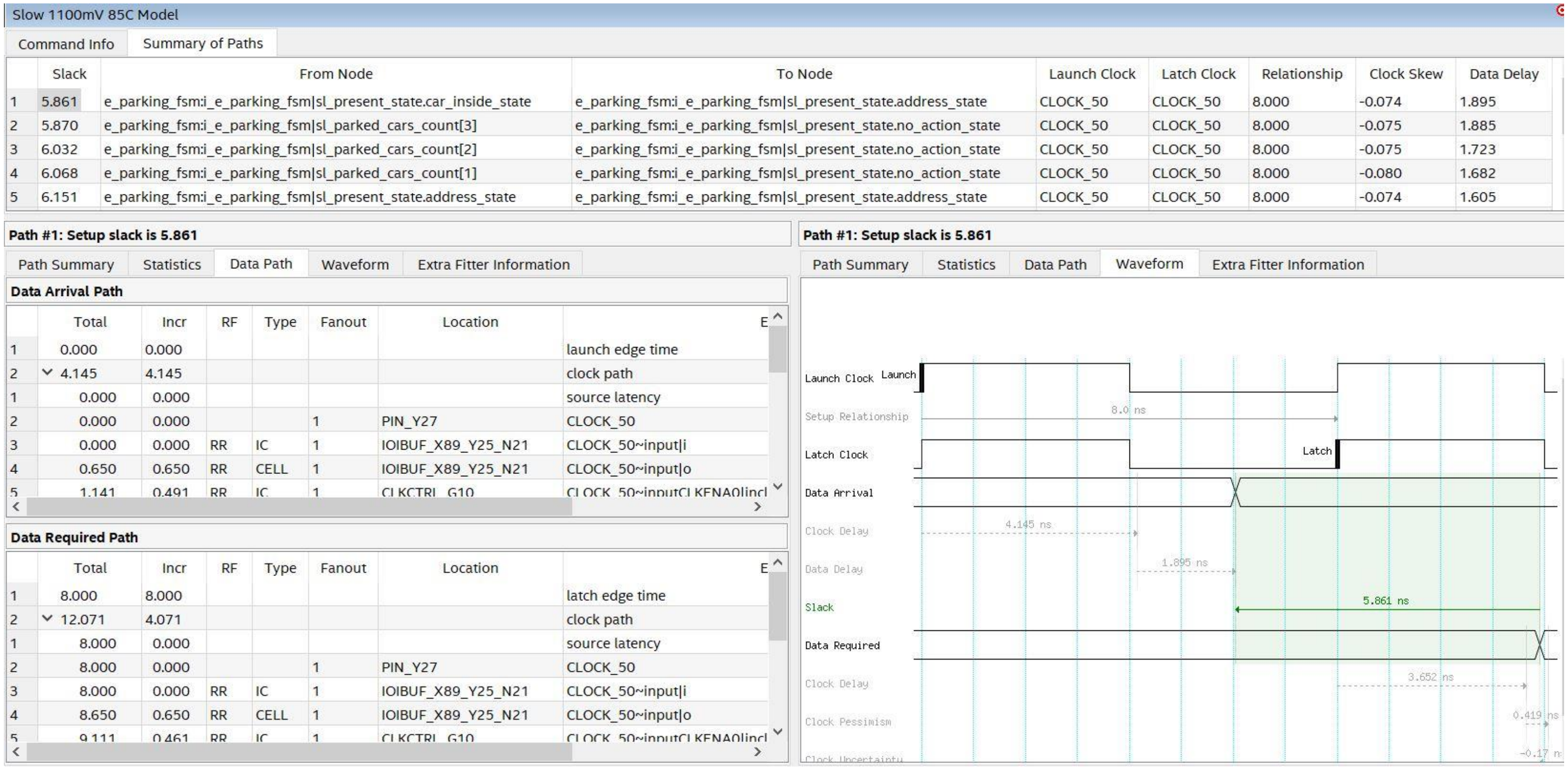
Figure 5 Fmax Summary Report



Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	CLOCK_50	0.700	0.000
2	e_payment_fsm:i_e_payment_fsm|sl_coin_state.one_euro_297	4.704	0.000

Figure 6 Positive Slack



Slow 1100mV 85C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	5.861	e_parking_fsm:i_e_parking_fsm|sl_present_state.car_inside_state	e_parking_fsm:i_e_parking_fsm|sl_present_state.address_state	CLOCK_50	CLOCK_50	8.000	-0.074	1.895
2	5.870	e_parking_fsm:i_e_parking_fsm|sl_parked_cars_count[3]	e_parking_fsm:i_e_parking_fsm|sl_present_state.no_action_state	CLOCK_50	CLOCK_50	8.000	-0.075	1.885
3	6.032	e_parking_fsm:i_e_parking_fsm|sl_parked_cars_count[2]	e_parking_fsm:i_e_parking_fsm|sl_present_state.no_action_state	CLOCK_50	CLOCK_50	8.000	-0.075	1.723
4	6.068	e_parking_fsm:i_e_parking_fsm|sl_parked_cars_count[1]	e_parking_fsm:i_e_parking_fsm|sl_present_state.no_action_state	CLOCK_50	CLOCK_50	8.000	-0.080	1.682
5	6.151	e_parking_fsm:i_e_parking_fsm|sl_present_state.address_state	e_parking_fsm:i_e_parking_fsm|sl_present_state.address_state	CLOCK_50	CLOCK_50	8.000	-0.074	1.605

Path #1: Setup slack is 5.861

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	E
1	0.000	0.000					launch edge time
2	⌄ 4.145	4.145					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1	PIN_Y27	CLOCK_50
3	0.000	0.000	RR	IC	1	IOIBUF_X89_Y25_N21	CLOCK_50~input|i
4	0.650	0.650	RR	CELL	1	IOIBUF_X89_Y25_N21	CLOCK_50~input|o
5	1.141	0.491	RR	IC	1	CLKCTRL_G10	CLOCK_50~inputCLKENA0|incl

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	E
1	8.000	8.000					latch edge time
2	⌄ 12.071	4.071					clock path
1	8.000	0.000					source latency
2	8.000	0.000			1	PIN_Y27	CLOCK_50
3	8.000	0.000	RR	IC	1	IOIBUF_X89_Y25_N21	CLOCK_50~input|i
4	8.650	0.650	RR	CELL	1	IOIBUF_X89_Y25_N21	CLOCK_50~input|o
5	9.111	0.461	RR	IC	1	CLKCTRL_G10	CLOCK_50~inputCLKENA0|incl

Path #1: Setup slack is 5.861

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Figure 7 Timing Requirements

6. Algorithmic State Machine

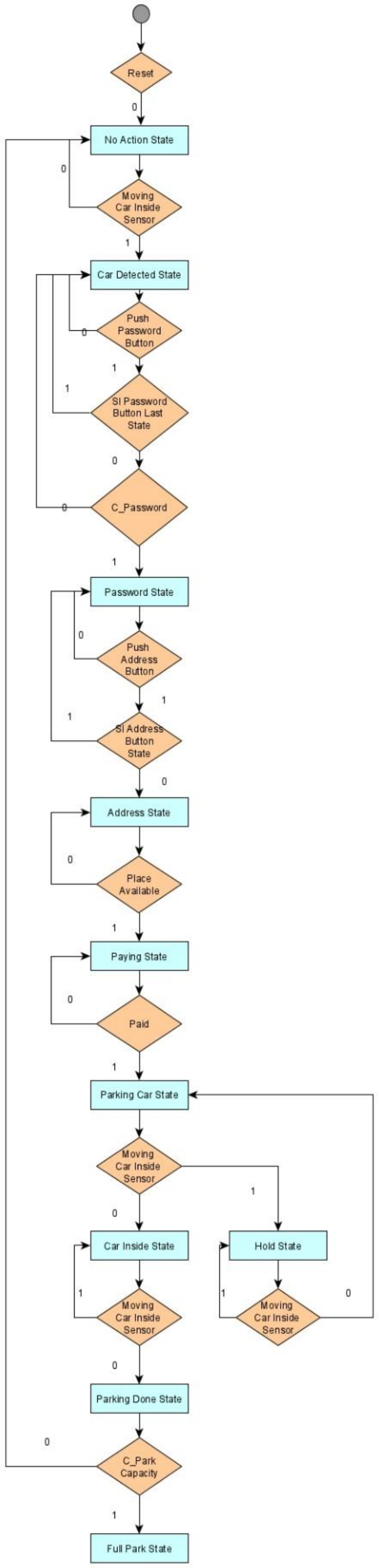


Figure 8 ASM Chart

7. Block Diagram Chart of Circuit

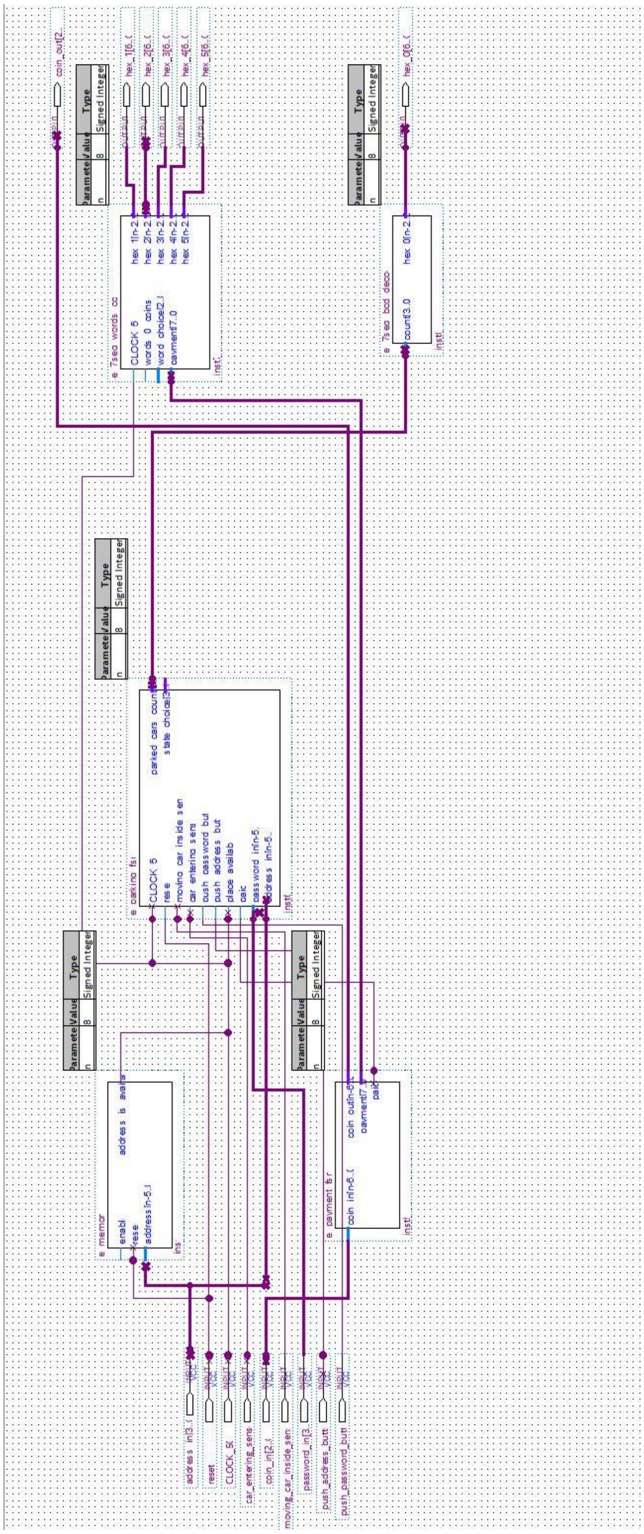


Figure 9 Block Diagram of Circuit