

APPROACH:

STEPS TAKEN TO REACH THE FINAL OUTPUT:

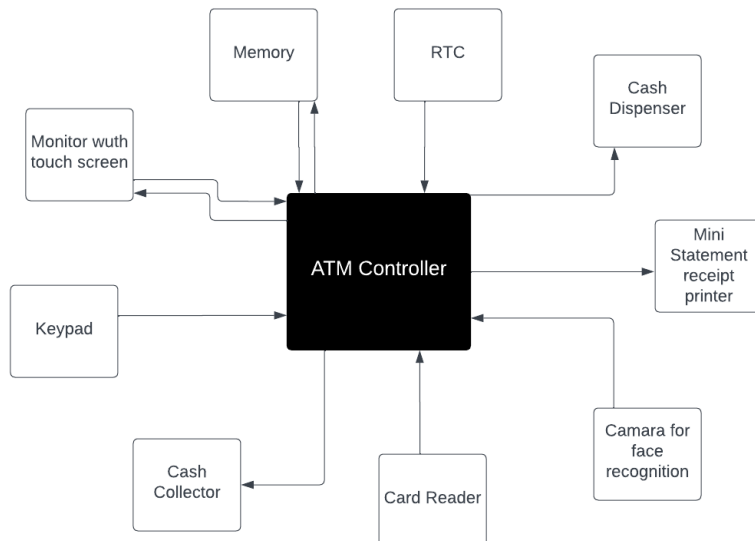
- a. Understanding the concepts
- b. Mealy state diagram
- c. Verilog code
- d. Verilog testbench
- e. Synthesis
- f. Fitter
- g. Assembler
- h. Timing analysis
- i. EDA Netlistviewer
- j. Clock divider
- k. Final Implementation

1.UNDERSTANDING THE CONCEPTS:

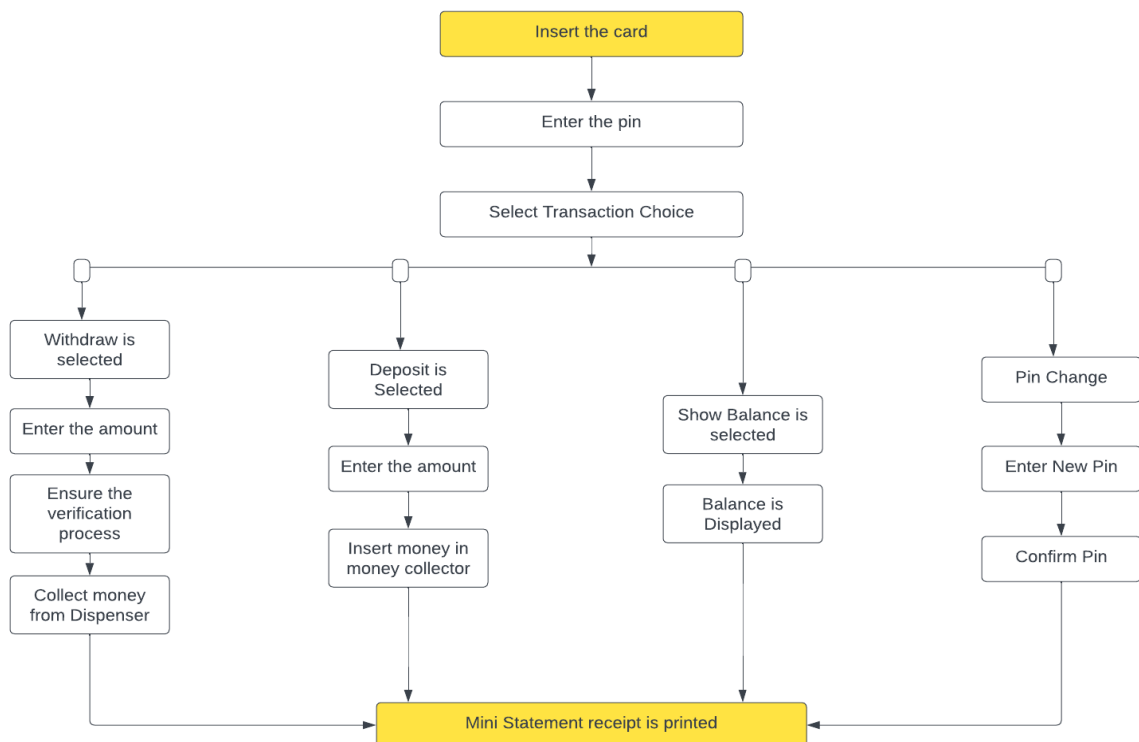
To begin with, comprehending the concepts, requirements, and specifications of an ATM controller is of paramount importance. This entails gaining a thorough understanding of how an ATM functions, the essential features it must incorporate, and the specific requirements and specifications set forth by the stakeholders. Understanding key concepts such as state diagrams, Mealy machines, and Moore machines plays a crucial role in the successful implementation of an ATM controller project.

2.MODEL BLOCK DIAGRAM:

Block diagrams offer a simplified and visual representation of complex systems, aiding in understanding, communication, and analysis during the design, development, and documentation phases of a project.

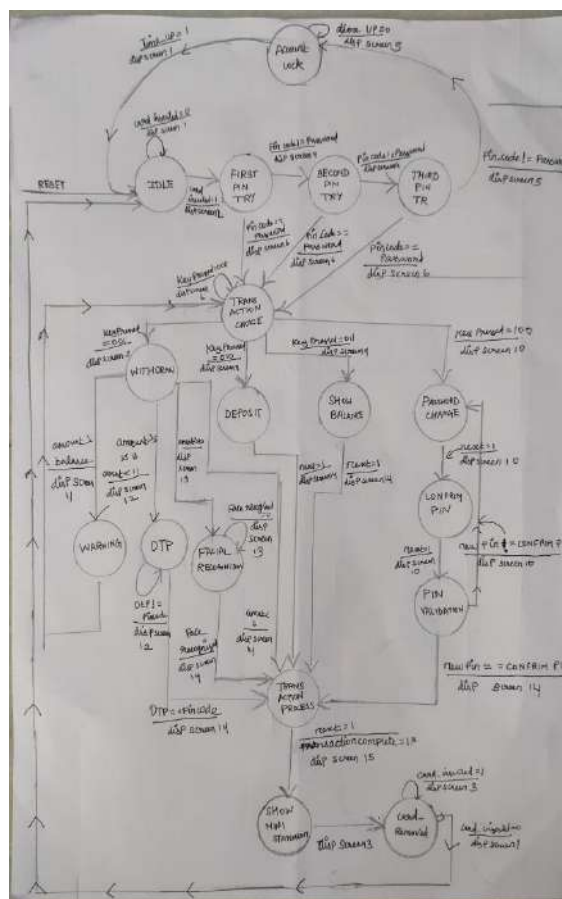


3.MODEL FLOWCHART:



4.Mealy state diagram:

State diagrams provide a visual representation of the different states that the ATM controller can be in and the transitions between those states. They help in identifying the various stages of an ATM transaction and the conditions under which the system transitions from one state to another. I prefer the Mealy state diagram as it allows for different outputs based on both the current state and the input signals. This flexibility enables the ATM controller to generate specific outputs that are dependent on the combination of the current state and the input received, making it suitable for handling the various transactions and producing appropriate outputs accordingly.



5. Verilog code:

Verilog was chosen as the hardware description language (HDL) for implementing the ATM controller. The FSM architecture was then translated into Verilog code, taking into account the necessary inputs, outputs, and internal variables.

Initially, a comprehensive set of inputs and outputs was defined to capture the different aspects and functionalities of the ATM controller. However, to adhere to the PIN constraints and optimize the design, the code was subsequently modified to minimize the number of inputs and outputs.

By carefully evaluating the requirements and considering the pin constraints, the code was refined to include only the essential inputs and outputs required for the proper functioning of the ATM controller. This approach helps to streamline the design, reduce complexity, and ensure efficient use of resources.

6.VERILOG TESTBENCH:

Several iterations of simulation and testing were performed to ensure the correctness of the design. Each functionality, such as PIN validation, withdrawal, deposit, balance display, and account lockout, was implemented and tested individually. Error conditions and exceptional cases were also considered during the testing phase.

Both the main program (ATM controller) and the testbench were thoroughly checked for syntax errors to ensure that the Verilog code is written correctly and adheres to the language rules and conventions. This involved carefully reviewing the code for any syntax mistakes, such as missing semicolons, incorrect variable declarations, or typographical errors.

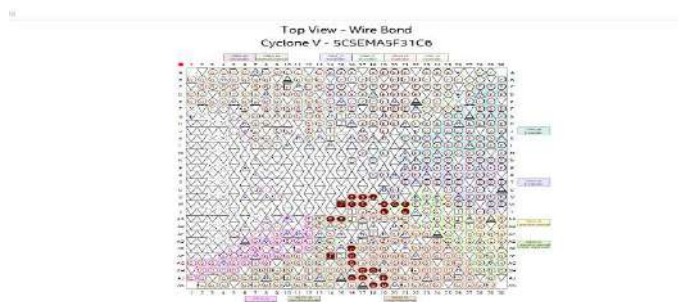
Additionally, simulation was performed using a MODELSIM to verify the behavior and functionality of the ATM controller. Test vectors and stimulus were provided to simulate different scenarios and validate the expected outputs and transitions between states.

7.SYNTHESIS:

The pins for all the inputs and outputs were accurately assigned by referring to the manual of the specific hardware platform being used, such as the Altera Cyclone V 5CSEMA5F31C6 device. Careful consideration was given to the pinout information provided in the manual to ensure that the correct pins were utilized for the respective signals.

After the pin assignments were made, synthesis was performed to transform the Verilog code into a gate-level representation that can be implemented on the target hardware device. During synthesis, the HDL code was analyzed, optimized, and mapped to the available resources of the FPGA.

The synthesis process involved mapping the high-level design constructs and logic elements to their corresponding low-level gate-level representations, such as lookup tables (LUTs), flip-flops, and multiplexers. The aim of synthesis was to generate an optimized netlist that can efficiently utilize the available resources of the FPGA.



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Analog Sense
card_inserted	Input	PIN_AF16	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault			
clk	Input	PIN_AF14	3B	B3B_N0	3.3-V LVTTTL		16mA ..ault			
disp_ne...ance[3]	Output	PIN_W20	5A	B5A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
disp_ne...ance[2]	Output	PIN_Y19	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
disp_ne...ance[1]	Output	PIN_W19	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
disp_ne...ance[0]	Output	PIN_W17	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
disp_ol...ance[3]	Output				2.5 V ..fault		12mA ..ault	1 (default)		
disp_ol...ance[2]	Output				2.5 V ..fault		12mA ..ault	1 (default)		
disp_ol...ance[1]	Output				2.5 V ..fault		12mA ..ault	1 (default)		
disp_ol...ance[0]	Output				2.5 V ..fault		12mA ..ault	1 (default)		
disp_pincod[1]	Output				2.5 V ..fault		12mA ..ault	1 (default)		
disp_pincod[0]	Output				2.5 V ..fault		12mA ..ault	1 (default)		
disp_screen[3]	Output	PIN_V18	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
disp_screen[2]	Output	PIN_V17	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
disp_screen[1]	Output	PIN_W16	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
disp_screen[0]	Output	PIN_V16	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
face_recognised	Input	PIN_W15	3B	B3B_N0	3.3-V LVTTTL		16mA ..ault			
key_pressed[2]	Input	PIN_AJ19	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault			
key_pressed[1]	Input	PIN_AK19	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault			
key_pressed[0]	Input	PIN_AK18	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault			
next	Input	PIN_AA15	3B	B3B_N0	3.3-V LVTTTL		16mA ..ault			
pin_code[1]	Input	PIN_AJ16	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault			
pin_code[0]	Input	PIN_AJ17	4A	B4A_N0	3.3-V LVTTTL		16mA ..ault			
reset	Input	PIN_AA14	3B	B3B_N0	3.3-V LVTTTL		16mA ..ault			
transac...omplete	Output	PIN_W21	5A	B5A_N0	3.3-V LVTTTL		16mA ..ault	1 (default)		
<<new node>>										

8.FITTER:

After the synthesis process, the next step in the design flow is the "fitting" or "place and route" stage. During this stage, the synthesized design netlist is mapped onto the physical resources of the target FPGA device. The fitting process determines the optimal placement of the logic elements (such as LUTs, flip-flops, and multiplexers) and routes the interconnections between them.


The fit report also includes information about the placement of the design's logic elements and the routing of the interconnections between them. It provides insight into how the design is mapped onto the physical FPGA device, helping to identify any potential issues or bottlenecks that may impact performance or timing.

Fitter Summary	
<<Filter>>	
Fitter Status	Successful - Sat Jul 15 05:24:42 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ATM_Controller
Top-level Entity Name	ATM_Controller
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	63 / 32,070 (< 1 %)
Total registers	21
Total pins	30 / 457 (7 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total RAM Blocks	0 / 397 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Fitter Resource Usage Summary

 <<Filter>>

	Resource	Usage	%
1	Logic utilization (ALMs needed / total ALMs on device)	63 / 32,070	< 1 %
2	▼ ALMs needed [=A-B+C]	63	
1	▼ [A] ALMs used in final placement [=a+b+c+d]	68 / 32,070	< 1 %
1	[a] ALMs used for LUT logic and registers	7	
2	[b] ALMs used for LUT logic	58	
3	[c] ALMs used for registers	3	
4	[d] ALMs used for memory (up to half of total ALMs)	0	
2	[B] Estimate of ALMs recoverable by dense packing	5 / 32,070	< 1 %
3	▼ [C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 32,070	0 %
1	[a] Due to location constrained logic	0	
2	[b] Due to LAB-wide signal conflicts	0	
3	[c] Due to LAB input limits	0	
4	[d] Due to virtual I/Os	0	
3			
4	Difficulty packing design	Low	
5			
6	▼ Total LABs: partially or completely used	9 / 3,207	< 1 %
1	-- Logic LABs	9	
2	-- Memory LABs (up to half of total LABs)	0	

 Find Next

M_Controller.v

Compilation Report - ATM_Controller


Fitter Resource Usage Summary


 <<Filter>>


	Resource	Usage	%
5			
6	▼ Total LABs: partially or completely used	9 / 3,207	< 1 %
1	-- Logic LABs	9	
2	-- Memory LABs (up to half of total LABs)	0	
7			
8	▼ Combinational ALUT usage for logic	111	
1	-- 7 input functions	0	
2	-- 6 input functions	17	
3	-- 5 input functions	20	
4	-- 4 input functions	14	
5	-- <=3 input functions	60	
9	Combinational ALUT usage for route-throughs	3	
10			
11	▼ Dedicated logic registers	21	
1	▼ -- By type:		
1	-- Primary logic registers	19 / 64,140	< 1 %
2	-- Secondary logic registers	2 / 64,140	< 1 %
2	▼ -- By function:		
1	-- Design implementation registers	19	

Fitter Resource Usage Summary

 <<Filter>>

	Resource	Usage	%
9	Combinational ALUT usage for route-throughs	3	
10			
11	▼ Dedicated logic registers	21	
1	▼ -- By type:		
1	-- Primary logic registers	19 / 64,140	< 1 %
2	-- Secondary logic registers 	2 / 64,140	< 1 %
2	▼ -- By function:		
1	-- Design implementation registers	19	
2	-- Routing optimization registers	2	
12			
13	Virtual pins	0	
14	▼ I/O pins	30 / 457	7 %
1	-- Clock pins	3 / 8	38 %
2	-- Dedicated input pins	0 / 21	0 %
15			
16	▼ Hard processor system peripheral utilization		
1	-- Boot from FPGA	0 / 1 (0 %)	
2	-- Clock resets	0 / 1 (0 %)	
3	-- Cross trigger	0 / 1 (0 %)	

 Find...

Fitter Resource Usage Summary			
 <<Filter>>			
	Resource	Usage	%
21	Total block memory implementation bits	0 / 4,065,280	0 %
22			
23	Total DSP Blocks	0 / 87	0 %
24			
25	Fractional PLLs	0 / 6	0 %
26	Global signals	1	
1	-- Global clocks	1 / 16	6 %
2	-- Quadrant clocks	0 / 66	0 %
3	-- Horizontal periphery clocks	0 / 18	0 %
27	SERDES Transmitters	0 / 100	0 %
28	SERDES Receivers	0 / 100	0 %
29	JTAGs	0 / 1	0 %
30	ASMI blocks	0 / 1	0 %
31	CRC blocks	0 / 1	0 %
32	Remote update blocks	0 / 1	0 %
33	Oscillator blocks	0 / 1	0 %
34	Impedance control blocks	0 / 4	0 %
35	Hard Memory Controllers	0 / 2	0 %
36	Average interconnect usage (total/H/V)	0.1% / 0.1% / 0.1%	

9.POWER REPORT:

The power report provides information on the power consumption of the device during the operation of the implemented design. It provides insights into the amount of power consumed by different components, such as logic elements, interconnects, I/O pins, and other resources of the FPGA.

- 218000 Using Advanced I/O Power to simulate I/O buffers with the specified board trace model
- 334003 Started post-fitting delay annotation
- 334004 Delay annotation completed successfully
- 215049 Average toggle rate for this design is 2.982 millions of transitions / sec
- 215031 Total thermal power estimate for the design is 422.16 mW
- Quartus Prime Power Analyzer was successful. 0 errors, 8 warnings

10.ASEMBLER:

Assembler generates the programming file with the extension.soc.That can be uploaded in the virtual lab setup such as lands land and we can view the result.

11.TIMING ANALYSIS:

Timing analysis enables us to analyze the timing behavior of a design and ensure that the desired timing requirements are met. Timing analysis plays a critical role in ensuring that the design operates reliably and within the desired performance limits.

iter.v Compilation Report - ARM_Controller

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

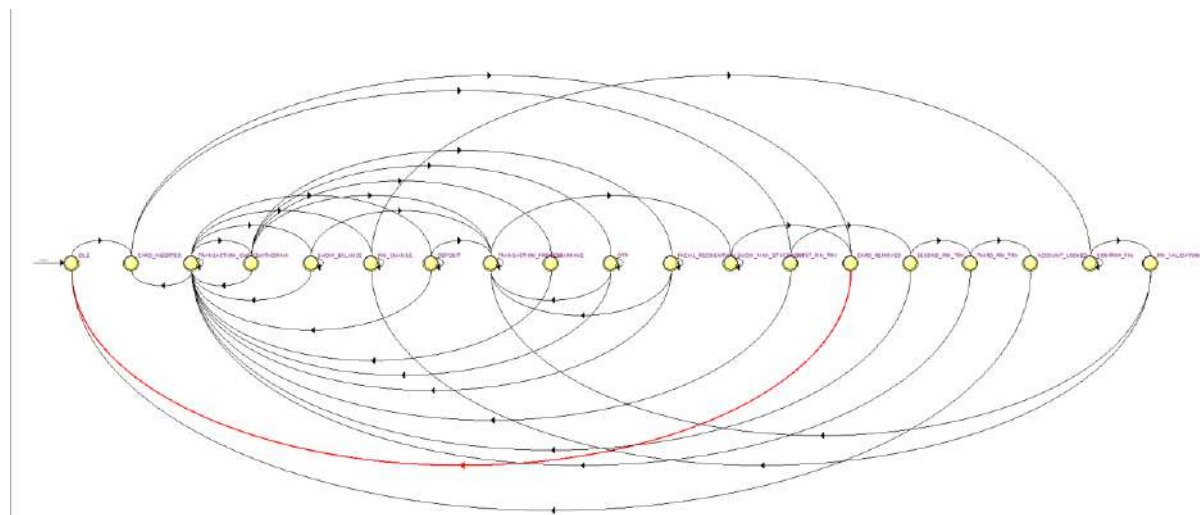
	Fmax	Restricted Fmax	Clock Name	Note
1	107.82 MHz	107.82 MHz	actual_clock	

- > i 332146 Worst-case setup slack is 13.840
- > i 332146 Worst-case hold slack is 0.172
- > i 332146 Worst-case recovery slack is 15.316
- > i 332146 Worst-case removal slack is 2.506
- > i 332146 Worst-case minimum pulse width slack is 9.067

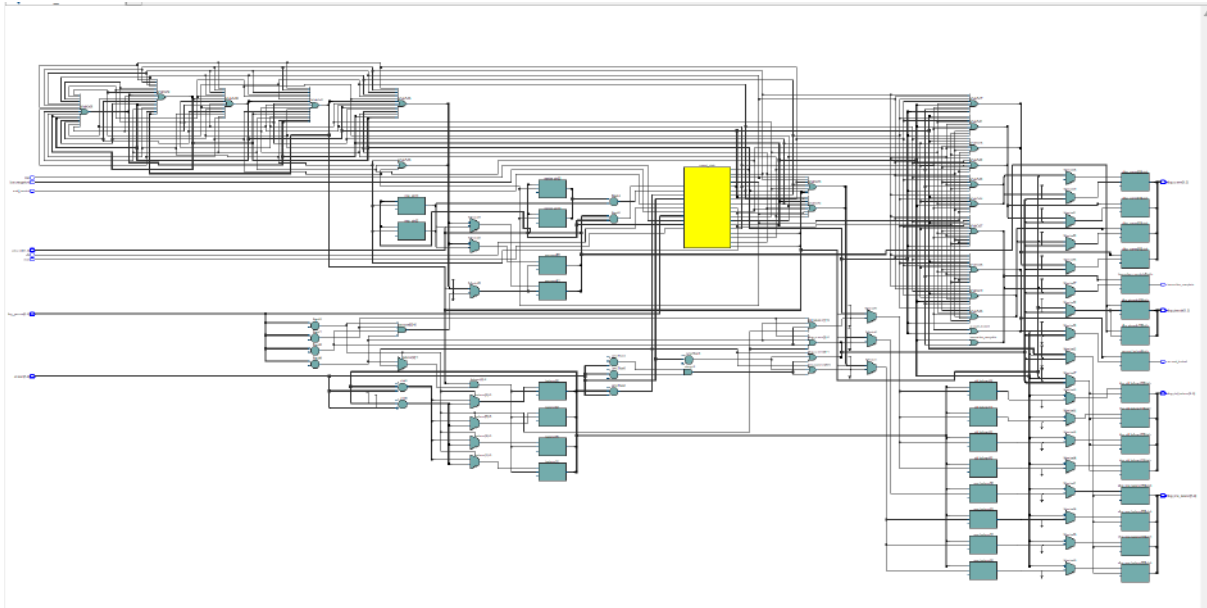
The input and outputs delays are given as 1ns and the clock cycle is given as 0-20ns with rise time 0 and fall time 10ns.

12.NETLIST VIEWER:

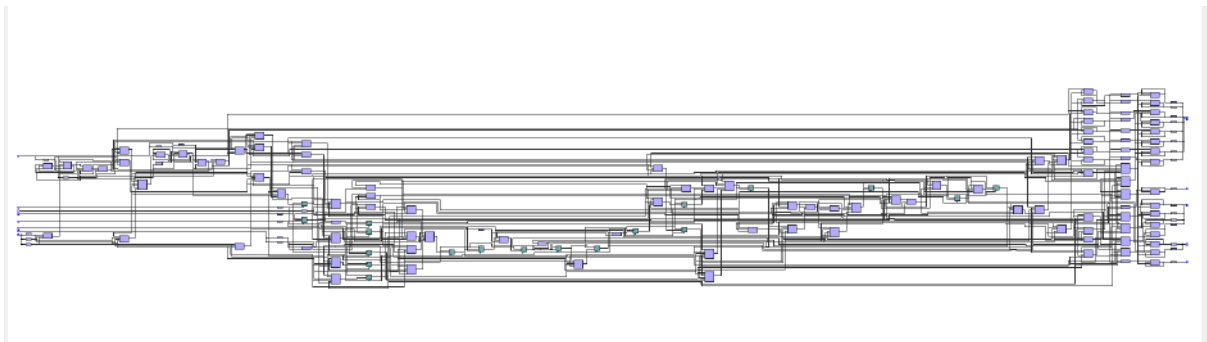
a.STATE DIAGRAM:



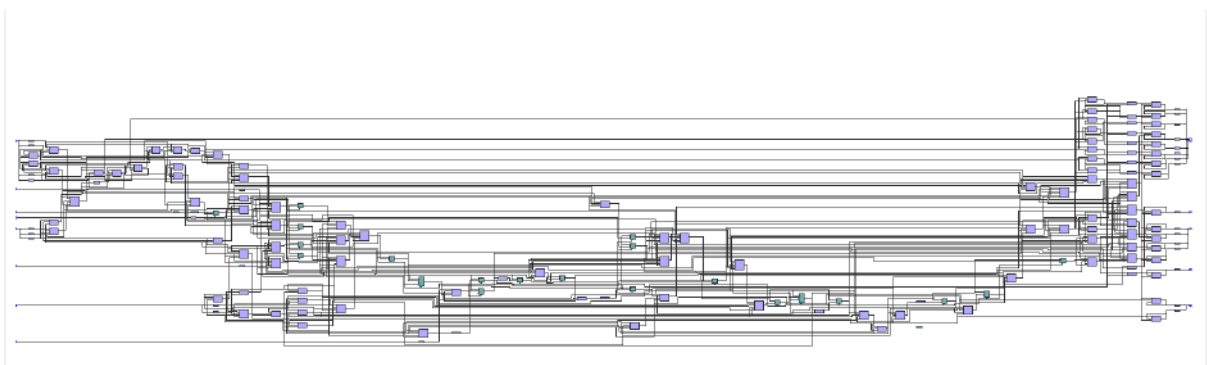
b.RTL VIEWER:



C.TECHNOLOGY MAP VIEWER (POST MAPPING):



d.TECHNOLOGY MAP VIEWER(POST FITTING):



13.CLOCK DIVIDER:

A clock divider is a digital circuit that takes an input clock signal and divides its frequency to generate a lower frequency output clock signal. The Clock_Divider

module takes an input clock `clk_in` with a frequency of 50 MHz and outputs a divided clock `clk_out`. This will enable us to view the outputs properly. Before using clock divider I was unable to see the outputs when the input is changed. So I implemented the clock divider. By dividing the input clock signal, it allows for slower clock cycles, providing more time for the outputs to stabilize and be observed on the board.

14.IMPLEMENTATION:



NOTE: After uploading the code in the board, I'm getting the first output and when I change the input I couldn't see any changes. I also tried the code which Intel group have sent (mealy 100 pattern detector) and I ended up facing the same issue.

So I tried to add clock divider and integrate that with the main code. But from Thursday night I'm facing server issues. Once the server is ready I will try to implement the design on the board and attach the outputs as soon as possible.