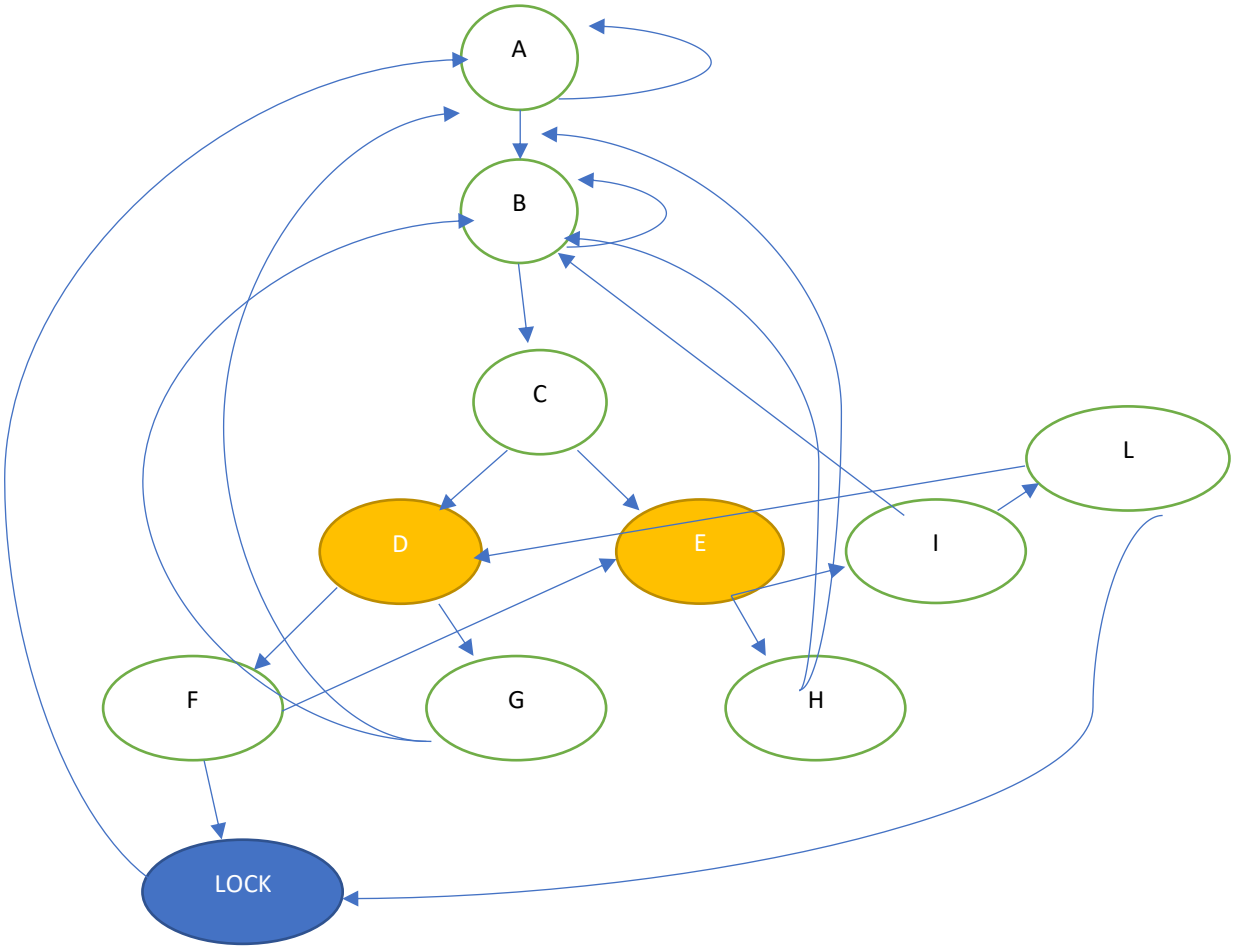
DIGITAL SYSTEM DESIGN APPLICATION PROJECT 2

In this project, a synchronous sequential circuit which detects two different 4-bit sequences A(0100) and B(0101) . A and B may overlap. When an A or a B word is detected, my circuit sets its Y output. My circuit go to a lock state, when two A or two B words (same words) are detected.

a=0100,b=0101;

STATE DIAGRAM:



STATE CODING

state	Binary code	X=0	X =1
A	0	A/0	B/0
B	01	C/0	B/0
C	010	E/1	D/1
D	0101	F/0	G/0
E	0100	H/0	I/0
F	01010	E/1	LOCK/1
G	01011	A/0	B/0
H	01000	A/0	B/0
I	01001	L/0	B/0
LOCK	010101		
L	010010	LOCK/1	D/1

RESET
A/0
A/0
A/0
A/0
A/0
A/0
A/0
A/0
A/0
A/0
A/0

I defined 0 for the initial value state A, as well as the reset state. Later, I defined a new state and a new output state for each value added to the x input, and I am sharing an example operation below .

X=000110100101.

It starts with A, comes 0, and stays in a state again. when a comes, it goes to state B. With the next one, it returns to the b state again, when it reaches the value I indicated with the arrow, it defines the number 4 and the output becomes 1

I'm using the meely machine. Because it has a function that depends on both the state and the input expression.

State encoding: Using a more compact encoding for states can reduce the number of flip-flops required to implement the FSM. For example, using binary encoding would require more flip-flops than using one-hot encoding for the same number of states.

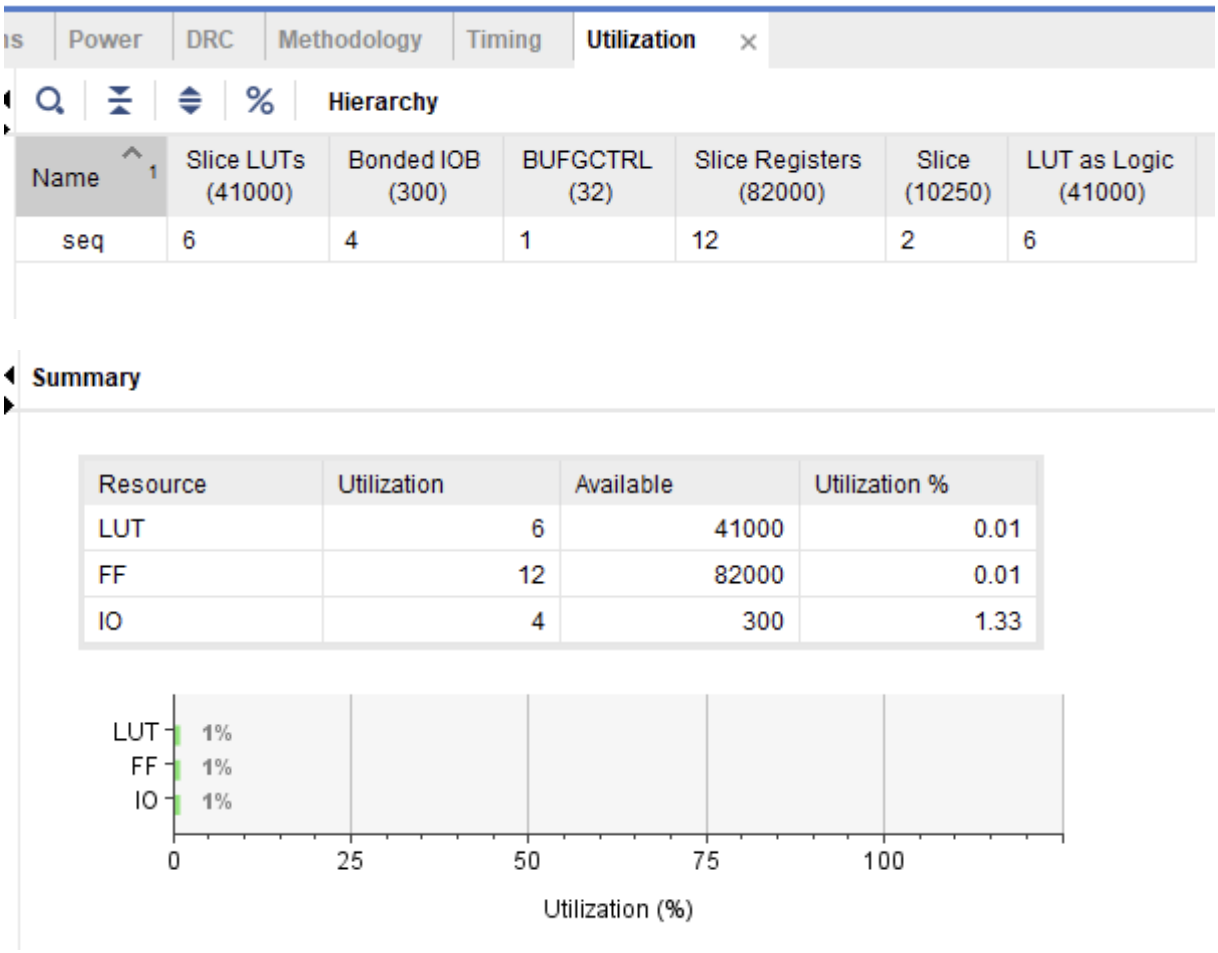
State assignment: Careful assignment of states can minimize the number of flip-flops needed to implement the FSM. For example, if two states are very similar, they can be merged into one state, reducing the number of flip-flops needed.

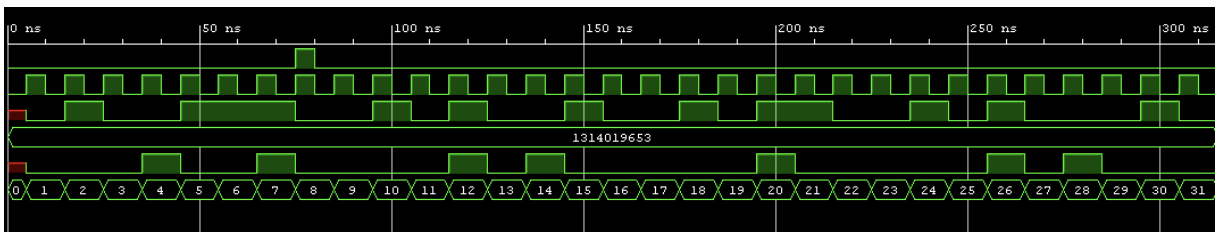
State minimization: State minimization algorithms can be used to identify and merge equivalent states in the FSM, reducing the total number of states. There are several algorithms available for this, such as the Quine-McCluskey method and the Karnaugh map method.

State encoding with don't-care terms: Using don't-care terms in state encoding can allow for the merging of states that would otherwise not be equivalent. For example, using binary encoding with don't-care terms can allow for the merging of states that differ only in the value of a don't-care bit.

FSM synthesis: FSM synthesis tools can be used to automatically generate an optimized FSM implementation from a high-level specification of the desired behavior. These tools can use a variety of techniques, such as state encoding, state assignment, and state minimization, to generate an efficient FSM implementation

In the Synthesis Settings menu, you can specify the type of synthesis tool you want to use (e.g. XST), and set various synthesis options such as optimization level, resource sharing, and maximum frequency. In the Implementation Settings menu, you can set various options such as the target device, clock frequency, and optimization level. You can also specify floorplanning constraints and timing constraints to help guide the placement and routing of your design. After setting the desired synthesis and implementation options, run synthesis and implementation to generate a netlist for your design. You can then use the Vivado Timing Analyzer to analyze the timing performance of your design and identify any timing violations. If there are timing violations, you can try adjusting your synthesis and implementation options, or adding additional timing constraints, to try and improve the timing performance.





5. I cant get y output. İf ı could this before output y=1 i get hazardous output.