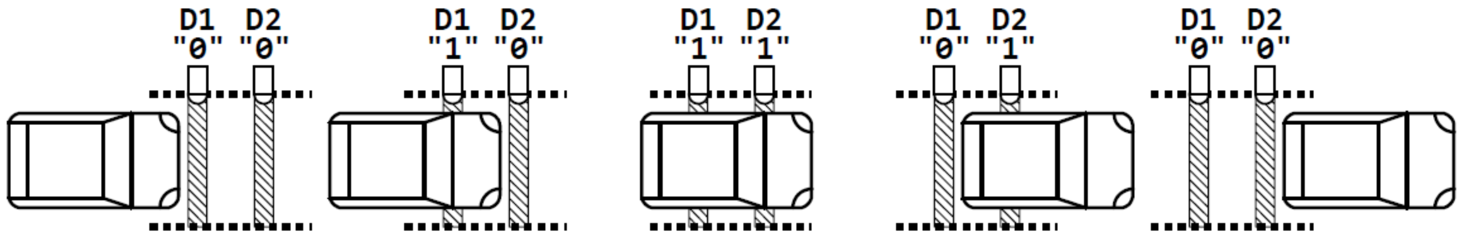
EE342 Lab-6
Instructions and Example Designs

Mehmet Çalı

Finite State Machine

- You need to design a finite state machine that counts the number of cars entering or exiting through a gate with two sensors.
- Sensors outputs high logic value if the car is detected, therefore in order for a car to be accounted pass through the gate input logic sequence should be as follows:



$$"00" \rightarrow "10" \rightarrow "11" \rightarrow "01" \rightarrow "00"$$

- Note that this is not the only valid sequence. In fact infinitely many sequences can be reproduced from the example above such as;

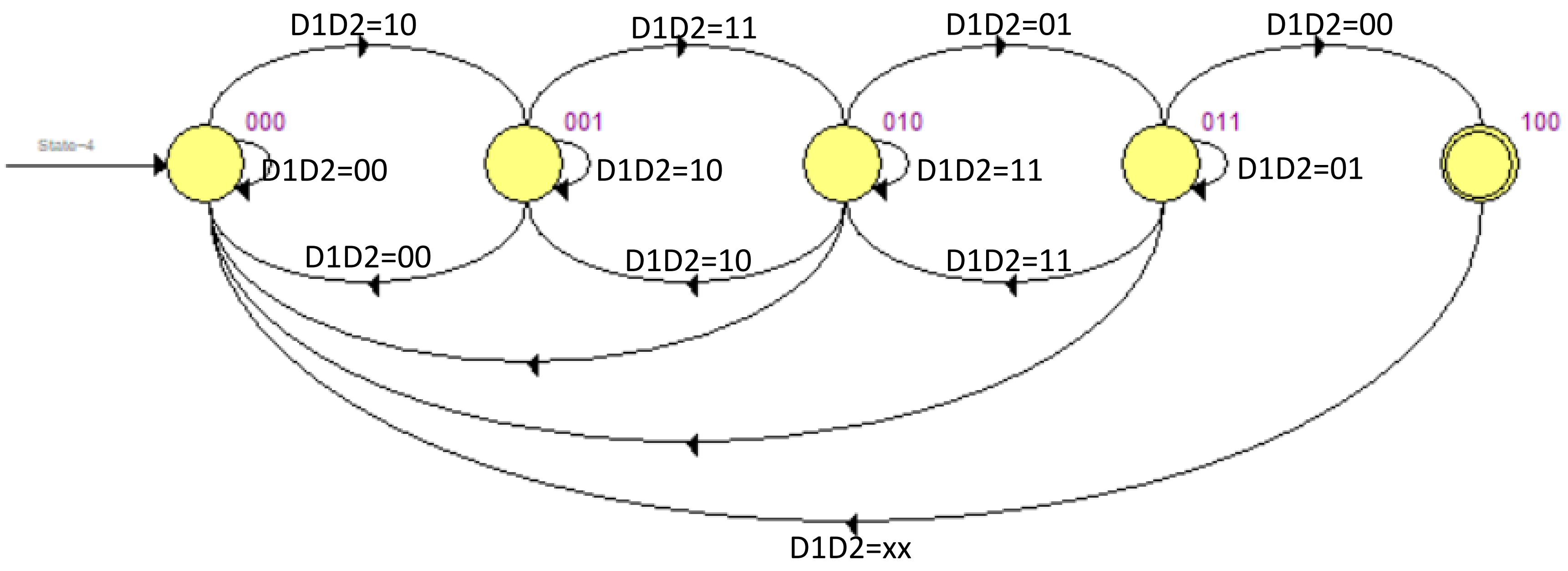
$$"00" \rightarrow "10" \rightarrow "11" \rightarrow "10" \rightarrow "11" \rightarrow "01" \rightarrow "00"$$

$$"00" \rightarrow "10" \rightarrow "00" \rightarrow "10" \rightarrow "11" \rightarrow "01" \rightarrow "00"$$

$$"00" \rightarrow "10" \rightarrow "11" \rightarrow "01" \rightarrow "11" \rightarrow "10" \rightarrow "11" \rightarrow "01" \rightarrow "00"$$

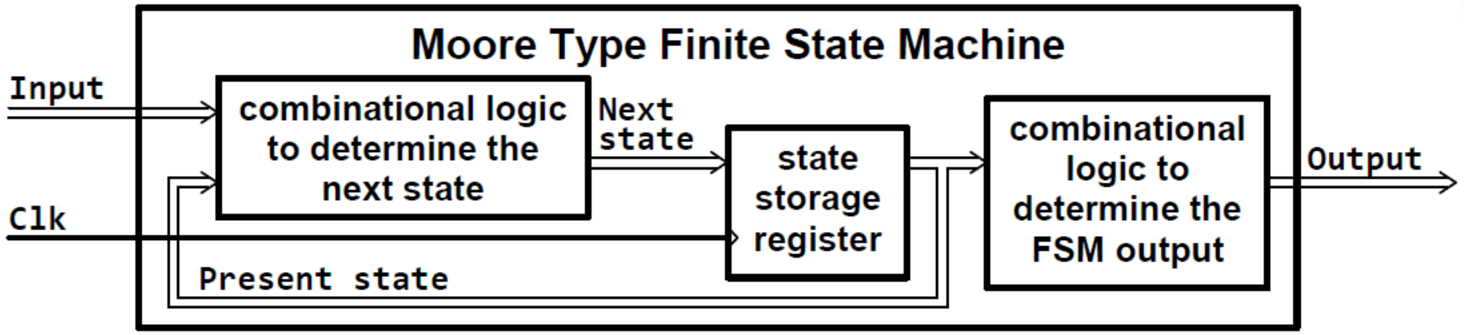
Finite State Machine

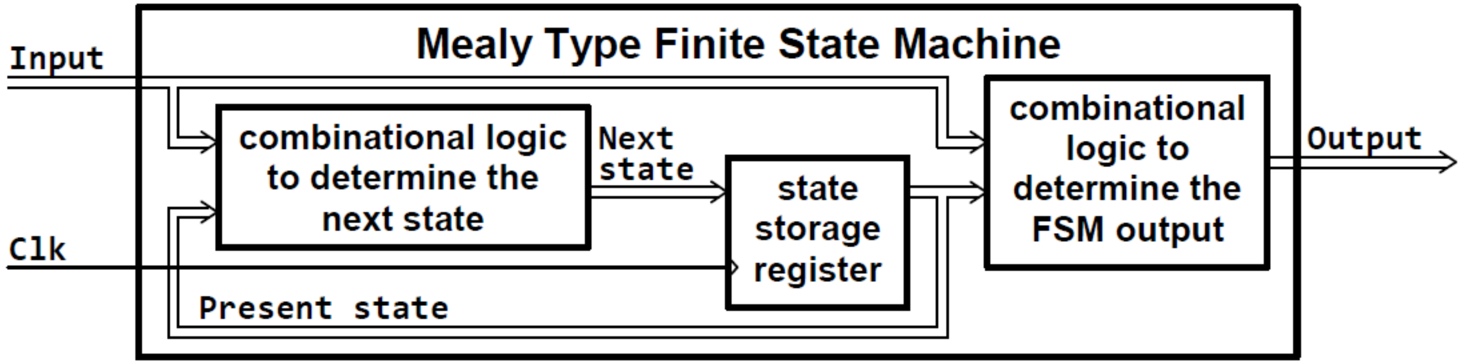
- In this experiment, in order to account all possible sequences we use finite state machines



Finite State Machine

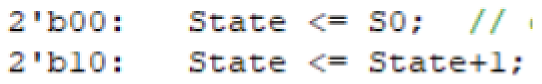
- While constructing your algoritm, you must use one of the following state machine architectures:


Moore Type Finite State Machine
Input
combinational logic to determine the next state
Next state
state storage register
combinational logic to determine the FSM output
Output
Clk
Present state


Mealy Type Finite State Machine
Input
combinational logic to determine the next state
Next state
state storage register
combinational logic to determine the FSM output
Output
Clk
Present state

Finite State Machine

- After synthesisizng your code you should check whether your design is detected as a finite state machine by Quartus or not. To do this you should double click Netlist Viewer >> State Machine Viewer and see if state machine is available.
- While writing your code you should assign default values to output, separate output logic from the state machine, use a reset condition that specifies the initial state for Quartus to detect your code as state machine.
- You should also avoid following arithmetic operations between states:



```
2'b00:    State <= S0;  //
2'b10:    State <= State+1;
```

Instead of this, explicitly specify state using:
2'b00: State <= S0; *or*
2'b00: State <= 000;

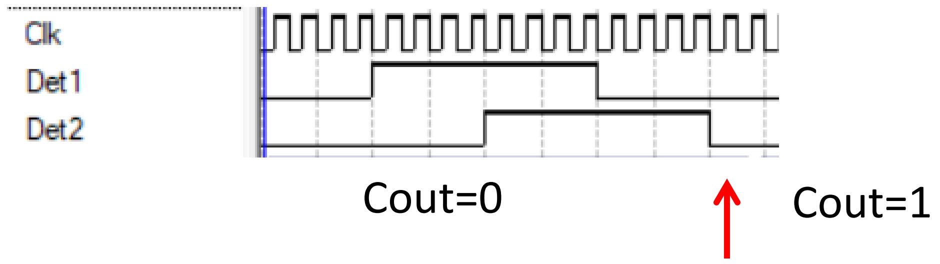
- If your code could not be detected as a state machine (if you cannot get rid of «Design has no State Machine» warning) even after you make sure that you apply the architecture in the previous slide and pay attention to issues mentioned in this slide, you should define both of the state encoding styles manually (will be described now).

Selecting Encoding Styles

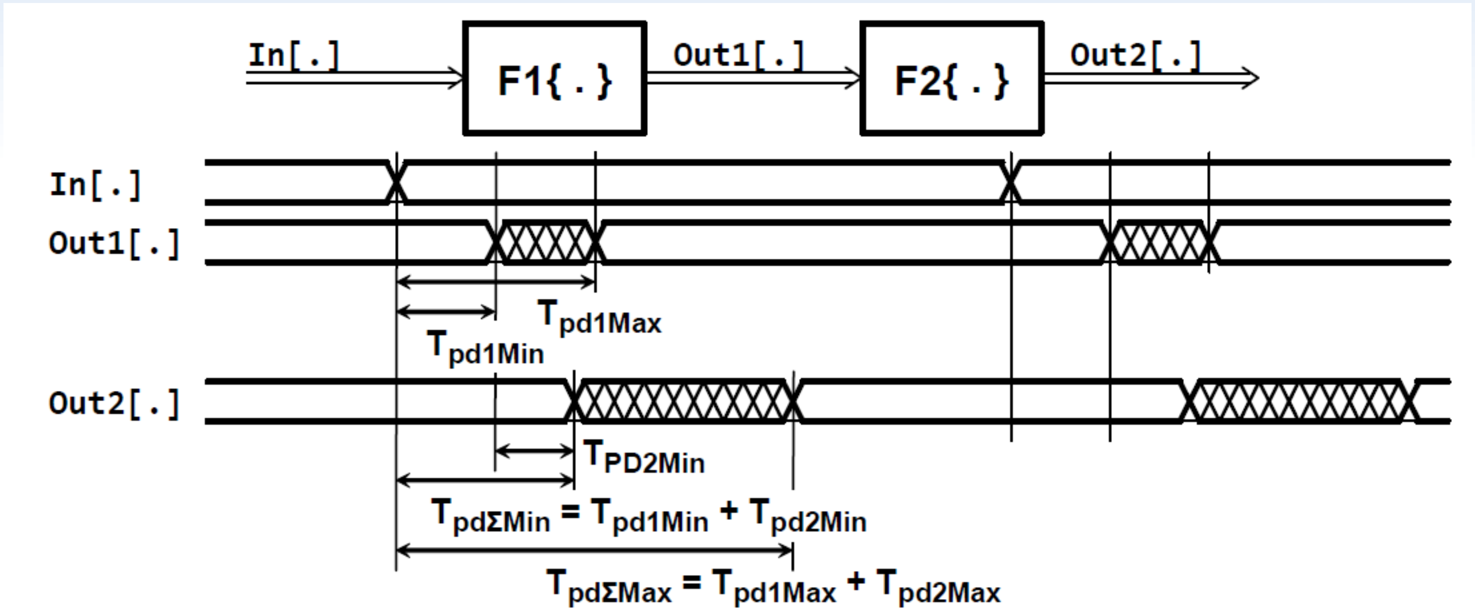
	If your code be detected as a state machine, follow these steps:	If your code cannot be detected as a state machine, follow these steps:
User Encoded	First you should define states appropirately and select User Encoded from Analysis & Synthesis Settings >>More Settings…>>State Machine Processing	The setting on the left does not mean anything since the system is not detected as a state machine. The states you define should directly be sythesized.
One-hot	In the second part you should select One-Hot from Analysis & Synthesis Settings >>More Settings…>>State Machine Processing You do not need any state value adjustment in your code	Again, the setting on the left does not mean anything so change your state values appropriately If you use parameters for state values and state size you can switch between One-Hot and User-Encoded easily
Checking Encoding Style Outputs	To check your encoding style, observe State Machine Viewer>>Encoding tab	To check your encoding style, you should observe the number of flip flop used in the Technology Map Viewer

Part-3

- We should detect both the cars entering and exiting in this part and count both of them.
- To do this instead of applying miscellaneous structures, you need to upgrade your state machine by adding states.
- You should check your design using various input combinations including the ones that involves cars that change their direction at various states.
- An example of car entering:

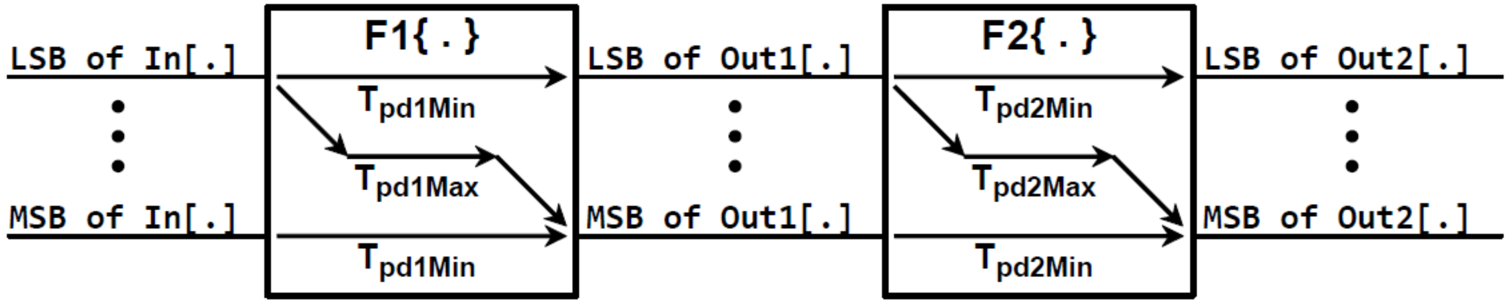


Timing of Cascaded Combinational Logic

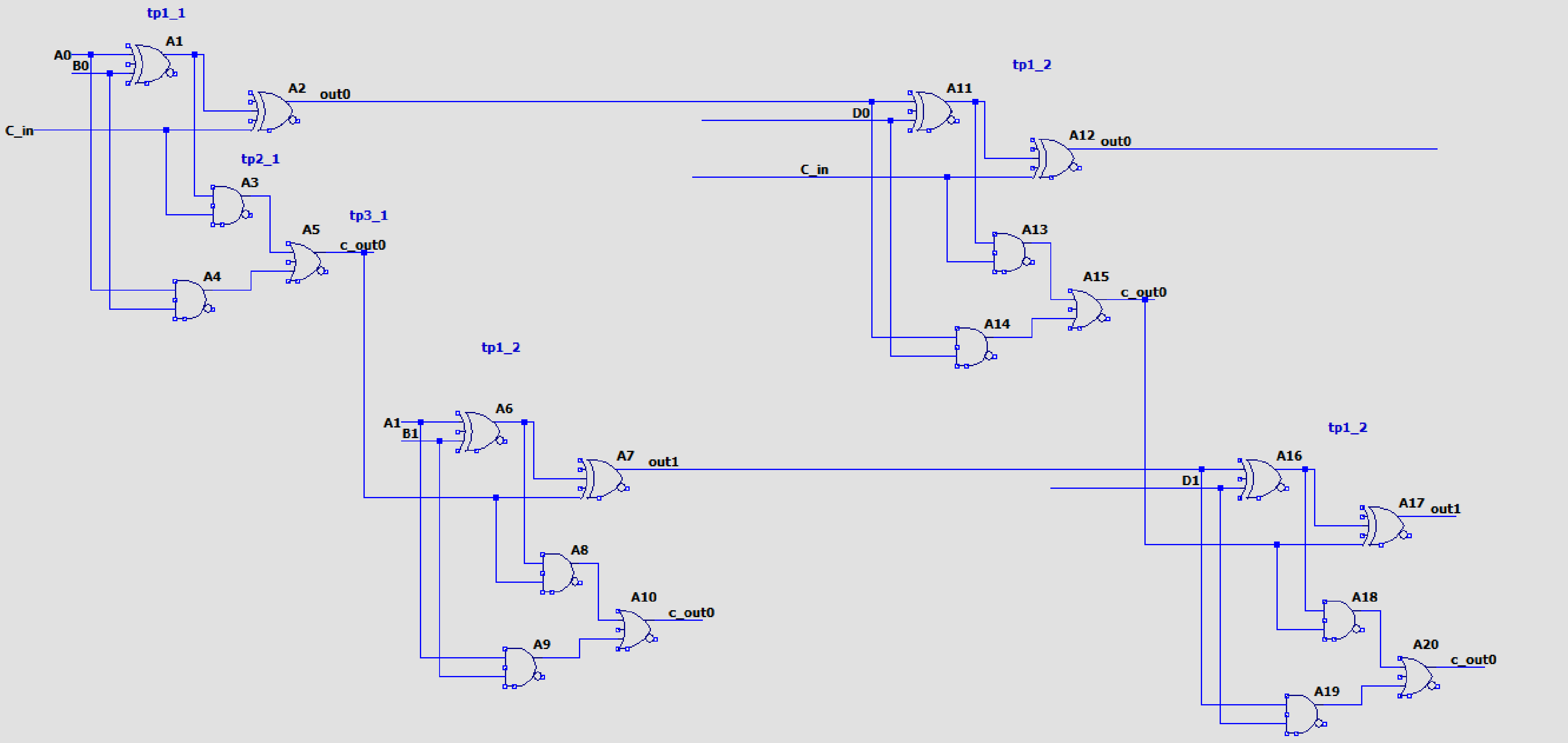


In[.]
F1{.}
Out1[.]
F2{.}
Out2[.]
In[.]
Out1[.]
T_{pd1Max}
T_{pd1Min}
Out2[.]
T_{PD2Min}
$T_{pd\Sigma Min} = T_{pd1Min} + T_{pd2Min}$
$T_{pd\Sigma Max} = T_{pd1Max} + T_{pd2Max}$

- For cascaded full adder:



Timing of Cascaded Full Adder



out0: (2tp1_1+2tp1_2)

out1: max(2tp1_1+tp1_2+tp2_2+tp3_2+tp1_2, tp1_1+tp2_1+tp3_1+tp1_1+2tp1_2)

tp_x_min=2tp1_x

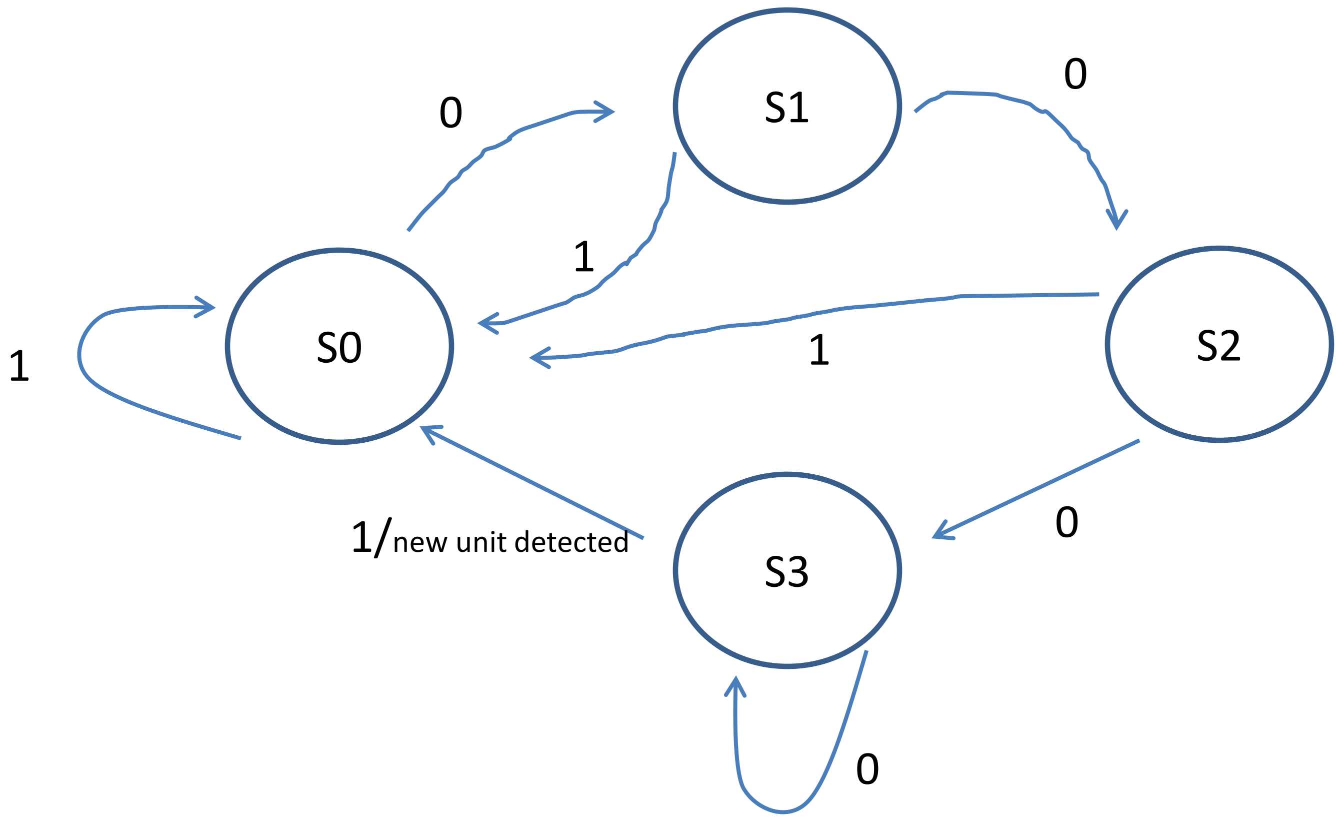
tp_x_max=tp1_x+tp2_x+tp3_x+tp1_x

Case Study: Data Unit Counter

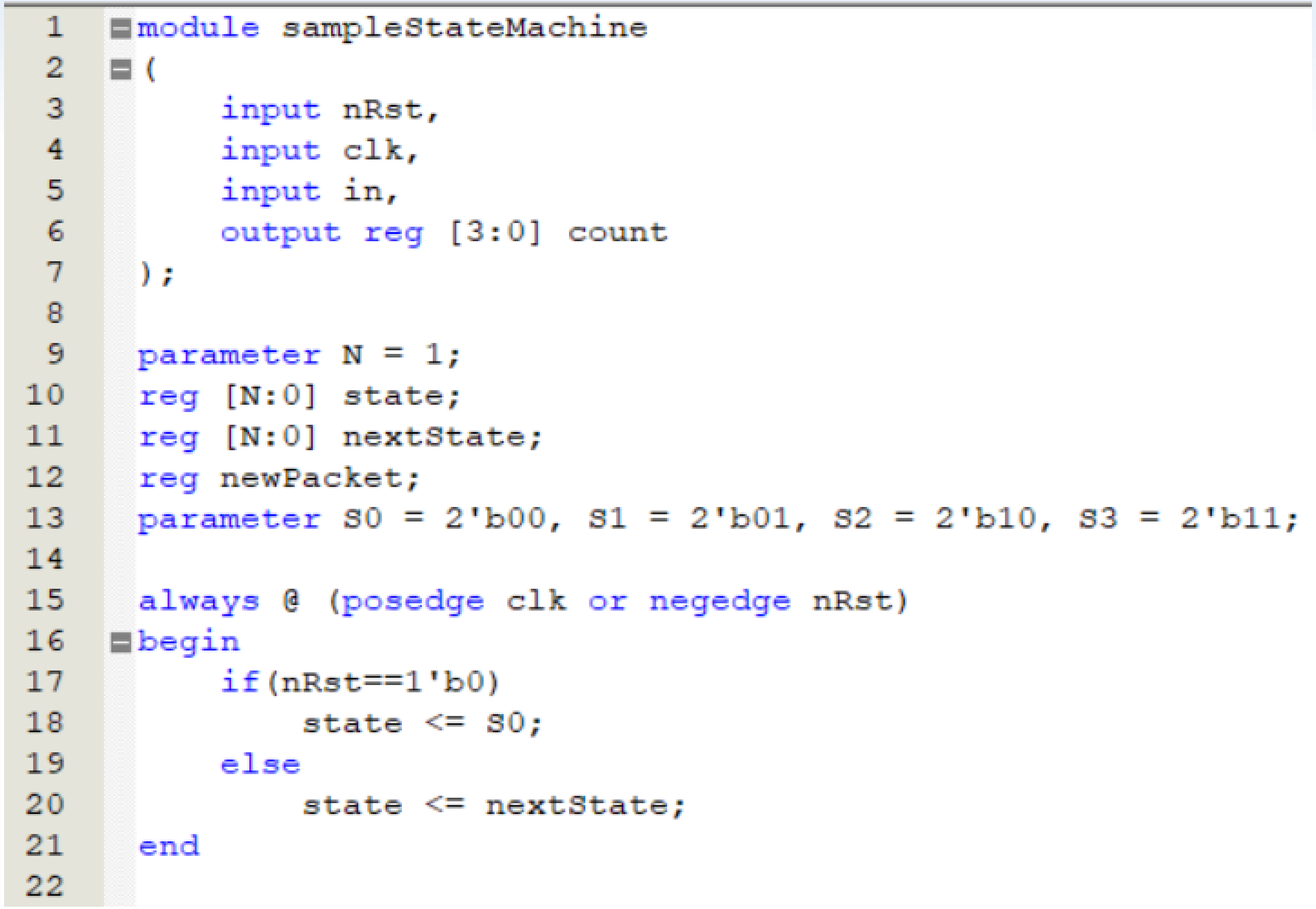
- Lets design a state machine of a structure that is constructed using an anology to how video packets are send and stored.
- Video files are stored in Network Abstraction Layer Units (NALU) in a way that a specified amount of NALU contained in each frame in the latest video coding standards. Each NALU can include variable amount of data (in bytes). In order to separate consequtive NALUs NALU prefixes are used (usually four byte data sequence of 0x00 0x00 0x00 0x01)
- Lets design a simplified version of such a structure
- Firstly, lets say we have a data source with single bit and it is consist of multiple units with variable length which are separated using a unit prefix (bit sequence of (0, 0, 0, 1))
- We want to calculate number of units in the data sequence using finite state machine (empty units should also be included)

Data Unit Counter

- The following flow chart represent the states to detect unit prefix of 0,0,0,1 in a data sequence

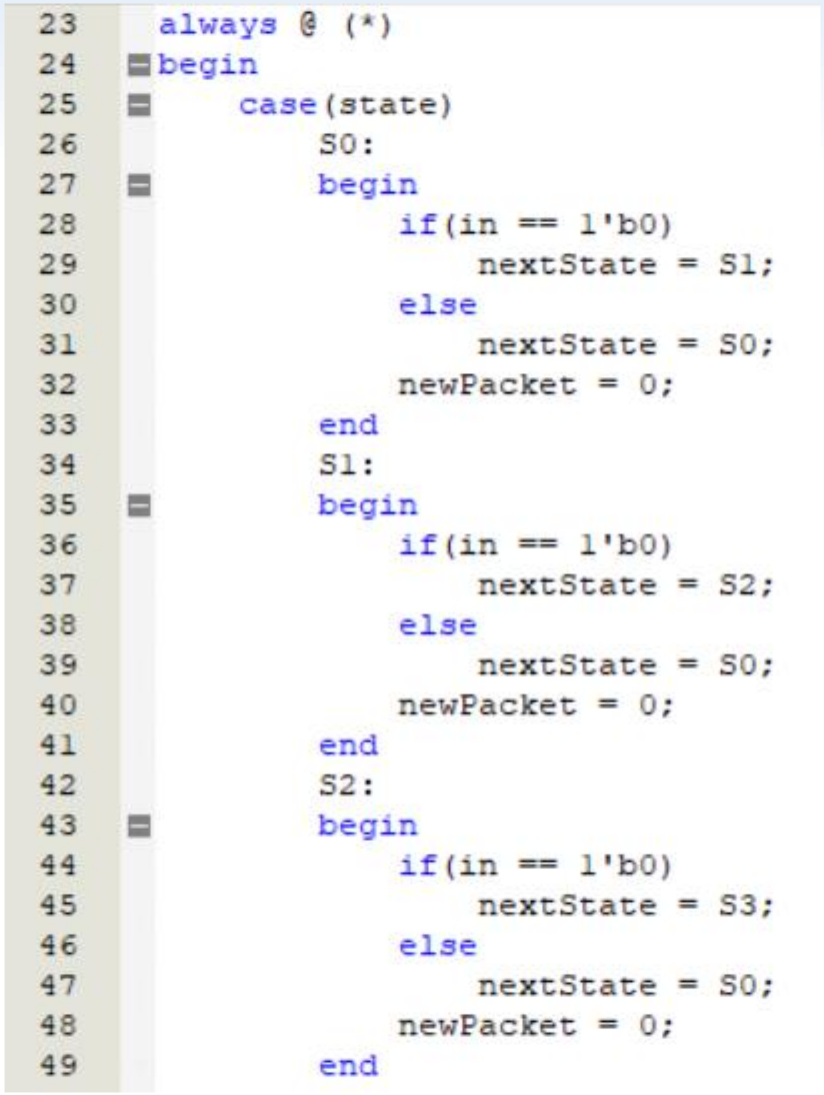


Data Unit Counter

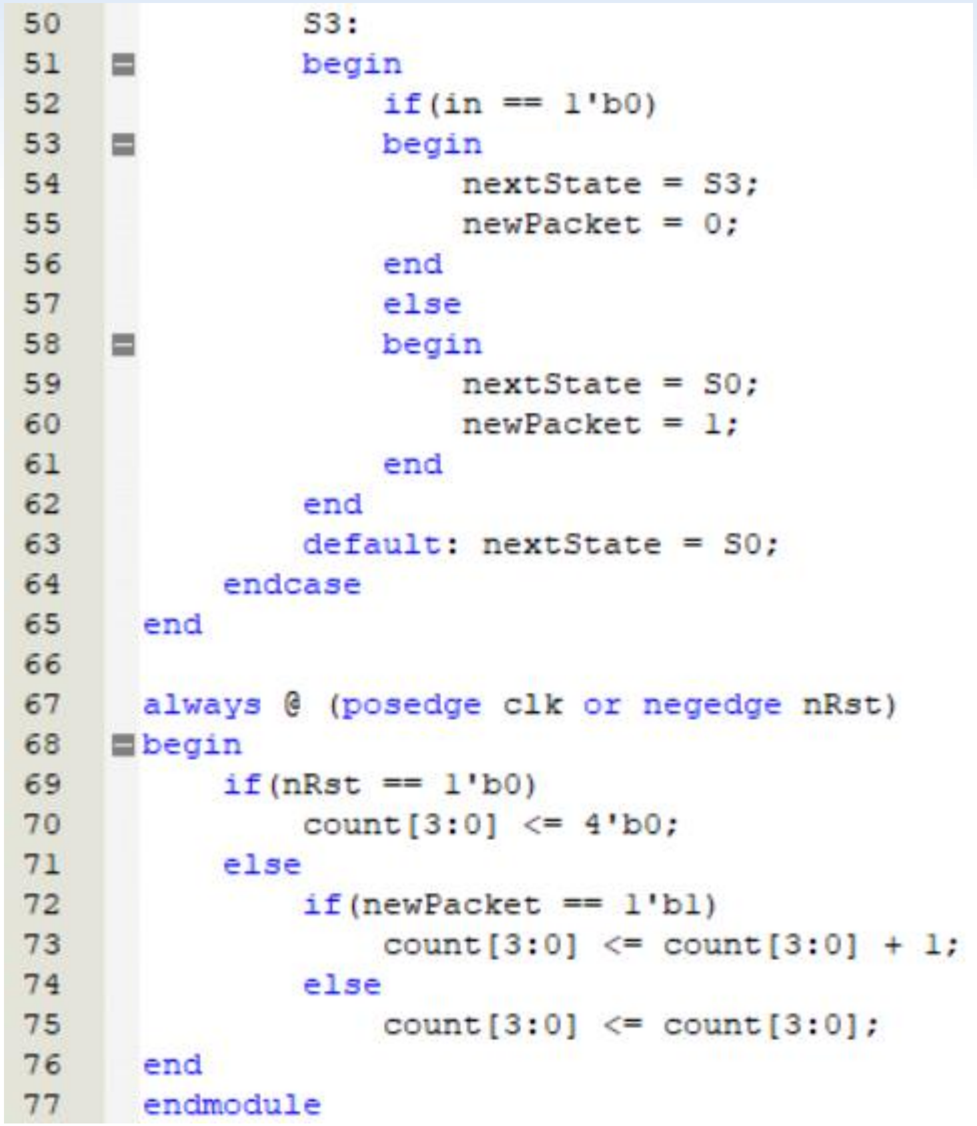


```verilog
module sampleStateMachine
(
	input nRst,
	input clk,
	input in,
	output reg [3:0] count
);

parameter N = 1;
reg [N:0] state;
reg [N:0] nextState;
reg newPacket;
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

always @ (posedge clk or negedge nRst)
begin
	if(nRst==1'b0)
		state <= S0;
	else
		state <= nextState;
end

```

Data Unit Counter

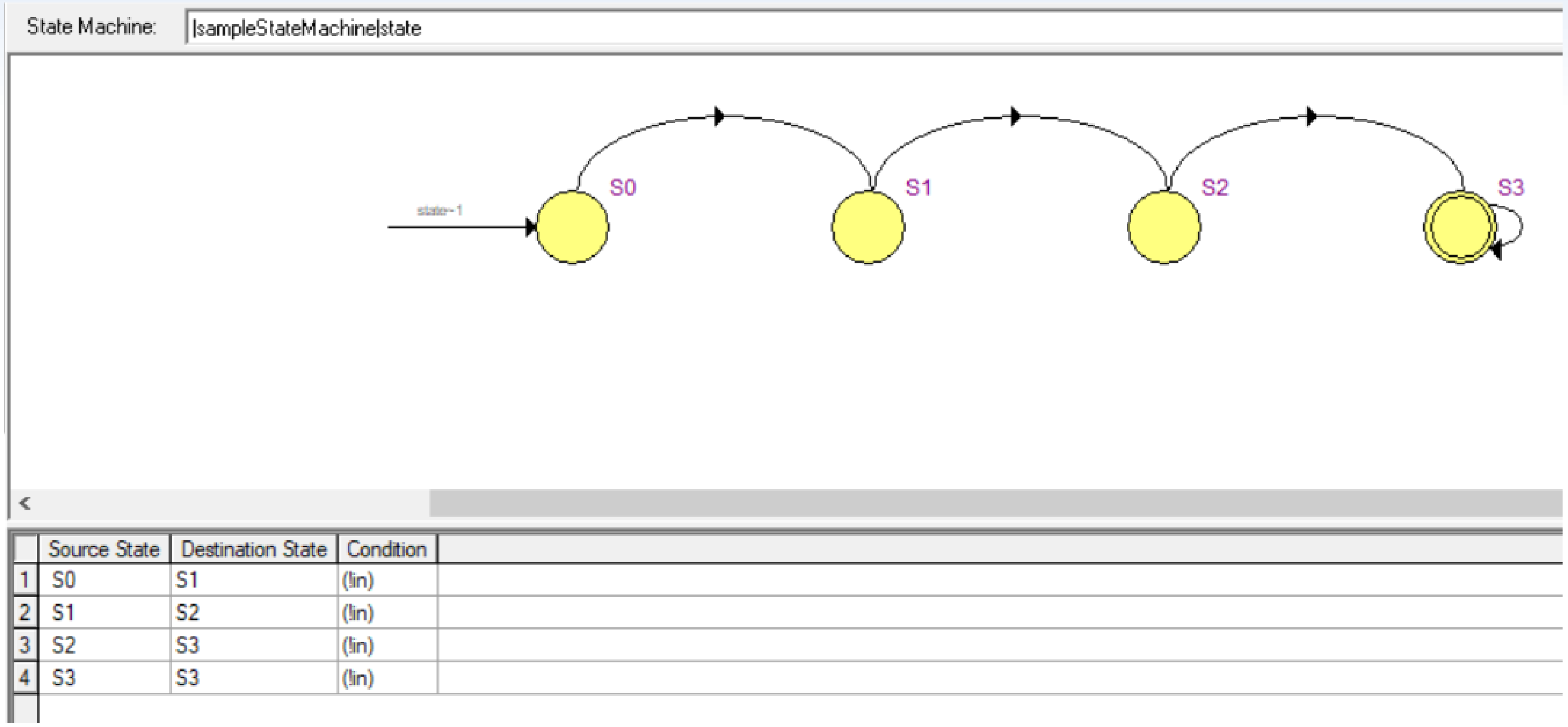


```
always @ (*)
begin
    case(state)
        S0:
        begin
            if(in == 1'b0)
                nextState = S1;
            else
                nextState = S0;
            newPacket = 0;
        end
        S1:
        begin
            if(in == 1'b0)
                nextState = S2;
            else
                nextState = S0;
            newPacket = 0;
        end
        S2:
        begin
            if(in == 1'b0)
                nextState = S3;
            else
                nextState = S0;
            newPacket = 0;
        end
```



```
        S3:
        begin
            if(in == 1'b0)
            begin
                nextState = S3;
                newPacket = 0;
            end
            else
            begin
                nextState = S0;
                newPacket = 1;
            end
        end
        default: nextState = S0;
    endcase
end

always @ (posedge clk or negedge nRst)
begin
    if(nRst == 1'b0)
        count[3:0] <= 4'b0;
    else
        if(newPacket == 1'b1)
            count[3:0] <= count[3:0] + 1;
        else
            count[3:0] <= count[3:0];
end
endmodule
```

State Machine Viewer

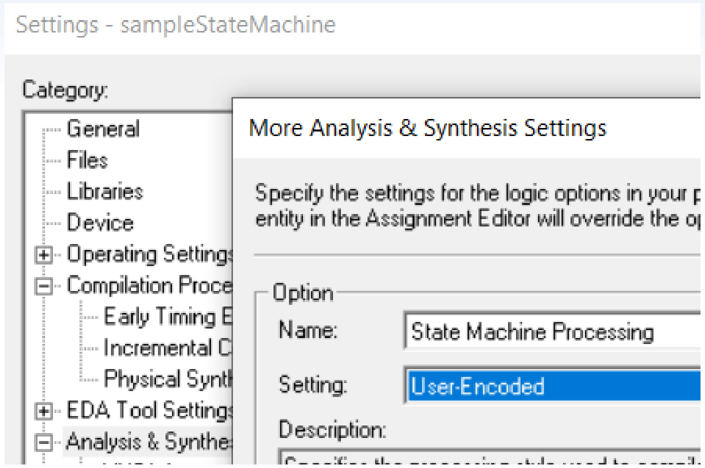


State Machine: |sampleStateMachine|state

	Source State	Destination State	Condition
1	S0	S1	(!in)
2	S1	S2	(!in)
3	S2	S3	(!in)
4	S3	S3	(!in)

Selecting Encoding Styles

- User Encoded Setting



Settings - sampleStateMachine

Category:

More Analysis & Synthesis Settings

Specify the settings for the logic options in your p
entity in the Assignment Editor will override the o

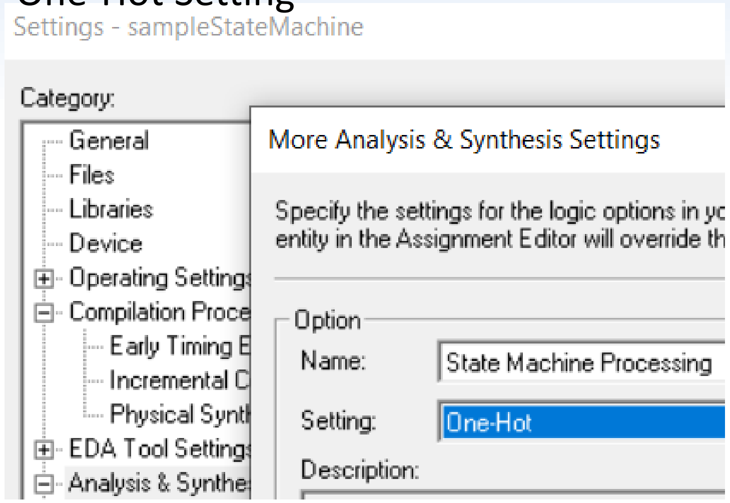
Option

Name: State Machine Processing

Setting: User-Encoded

Description:

- One-Hot Setting



Settings - sampleStateMachine

Category:

More Analysis & Synthesis Settings

Specify the settings for the logic options in yo
entity in the Assignment Editor will override th

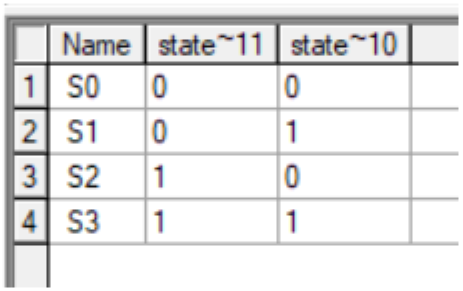
Option

Name: State Machine Processing

Setting: One-Hot

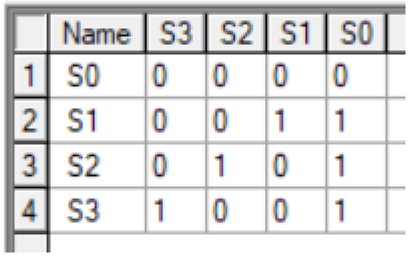
Description:

- User Encoded Output Encoder Table



	Name	state~11	state~10
1	S0	0	0
2	S1	0	1
3	S2	1	0
4	S3	1	1

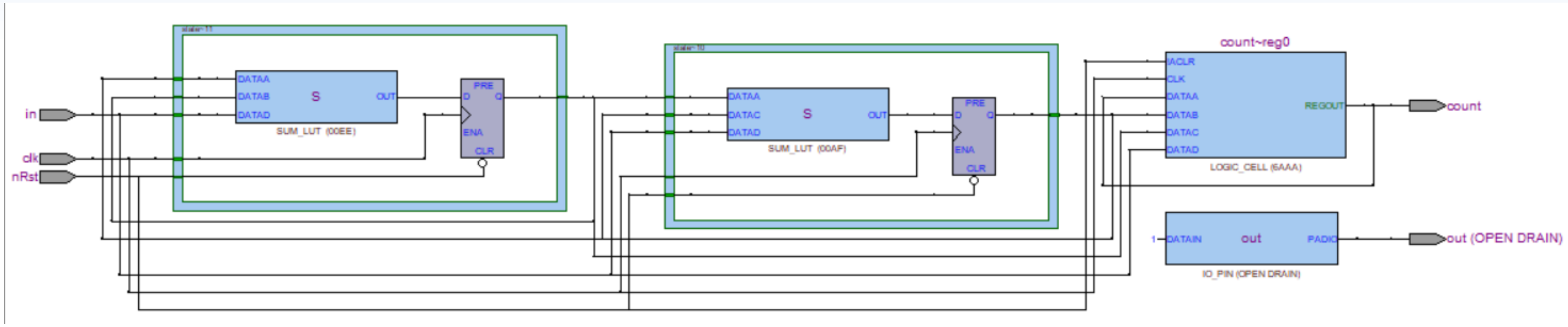
- One-Hot Output Encoder Table



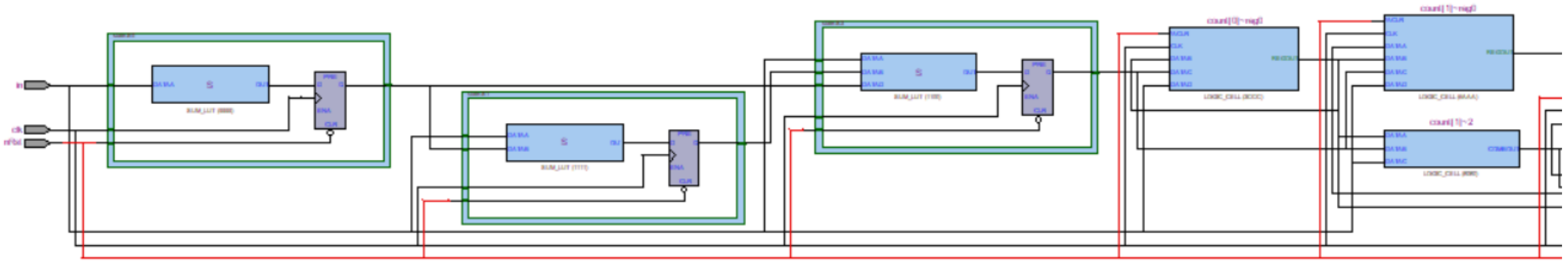
	Name	S3	S2	S1	S0
1	S0	0	0	0	0
2	S1	0	0	1	1
3	S2	0	1	0	1
4	S3	1	0	0	1

Selecting Encoding Styles

User Defined:



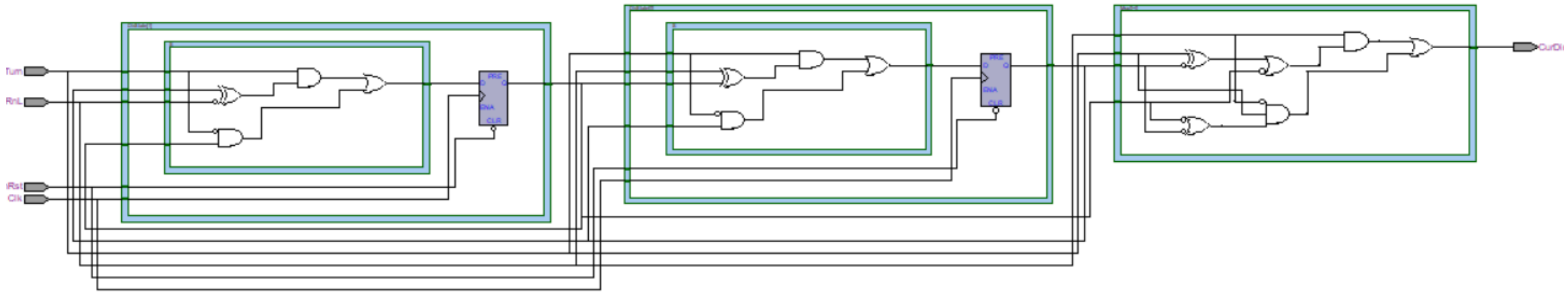
One Hot:



Selecting Encoding Styles

- Example of more general synthesizer outputs for a state machine with 4 states (there are no systhesized away states):

User Defined:



One Hot: