State Machine Design

Last updated 2/23/21

State Machine Design

- These slides walk through the process to design a finite state machine by hand

- Upon completion: You should be able to design a FSM by hand

State Machine Design

- Design Process

1) Identify the states – collectively these make a state variable
2) Identify the Inputs and Outputs
3) Assign values for each input/output (encoding)
4) Create a state transition diagram / table
5) Assign values for the state variable for each state (encoding)
6) Create truth tables for the combinational logic blocks in the machine model: next state, output
7) Minimize the next state and output equations using K-maps or Boolean Algebra techniques
8) Draw the circuit schematic
9) Verify the solution
10) Build the physical circuit
11) Test the physical circuit to ensure correct operation

State Machine Design

- Design Process – Identify the states

1
  - Most critical part of the process
  - Decisions here impact the complexity, speed, efficiency and cost of the solution
  - Focus on required outputs / actions

- Stop Light
  - RG, RY, GR, YR
  - Leaving out RR, Flashing Red, emergency override, …

State Machine Design

- Design Process – Identify the inputs and outputs

2
  - Driven by specification and state selection

- Stop Light
  - Inputs
    - Reset
    - Traffic N/S
    - Traffic E/W

  - Outputs
    - Green
    - Yellow
    - Red

State Machine Design

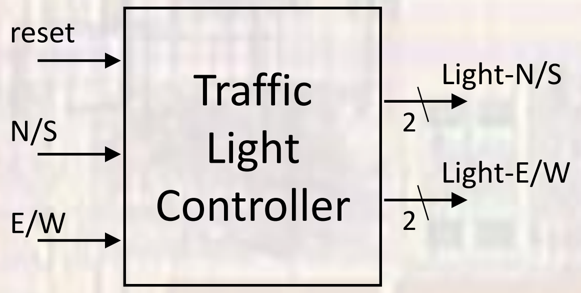
- Design Process – Assign values for each input/output (encoding)

3

- Stop Light
  - Inputs
    - Reset
    - Traffic N/S
    - Traffic E/W
  - Outputs
    - Green
    - Yellow
    - Red

Input	Encoding
Reset	n/a
N/S	n/a
E/W	n/a

Output	Encoding
Green	00
Yellow	01
Red	10



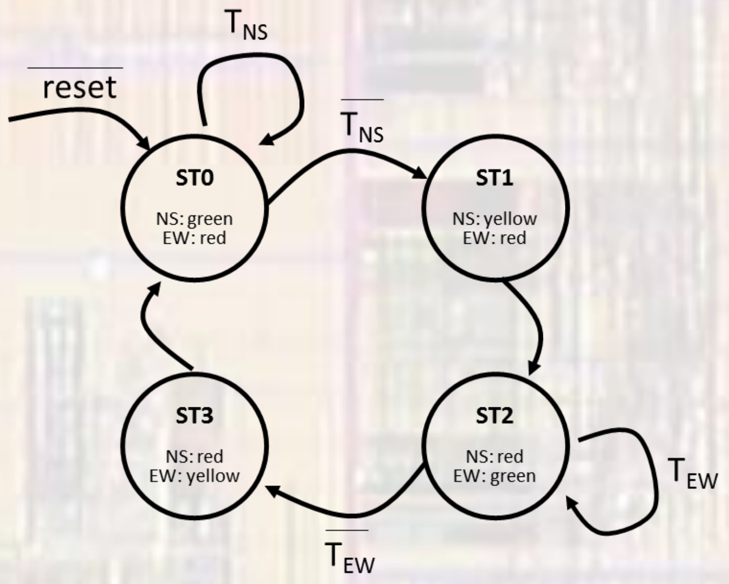
State Machine Design

- Design Process – Create a state transition diagram

4

  - Ensure each input and output is covered
  - Watch for possible infinite loops
  - Watch for floating (not accessible) states

- Stop Light

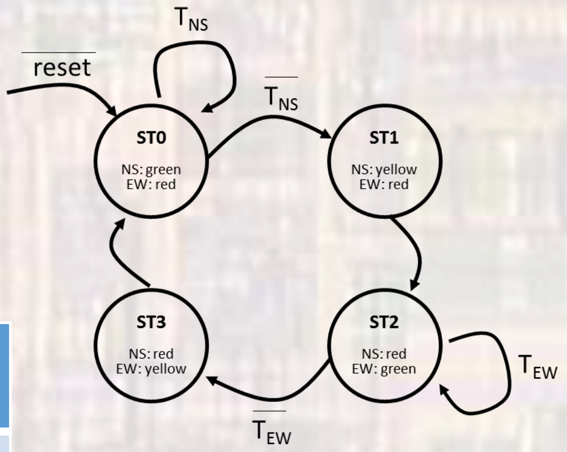


State Machine Design

- Design Process – Create a state transition table

4

- Ensure each input and output is covered
- Watch for possible infinite loops
- Watch for floating (not accessible) states



- Stop Light

Current State S	Inputs T_{NS}	T_{EW}	Next State S'
ST0	0	0	ST1
ST0	0	1	ST1
ST0	1	0	ST0
ST0	1	1	ST0

Current State S	Inputs T_{NS}	T_{EW}	Next State S'
ST0	0	x	ST1
ST0	1	x	ST0
ST1	x	x	ST2
ST2	x	0	ST3
ST2	x	1	ST2
ST3	x	x	ST0

State Machine Design

- Design Process – Assign state variable values for each state

5

  - Many possible encodings (covered later)
  - Simple solution is a binary encoding: 000, 001, 010, 011, …

- Stop Light
  - 4 states → 2 bit binary encoding

State	Encoding $S_{1:0}$
ST0	00
ST1	01
ST2	10
ST3	11

State Machine Design

- Design Process – Create truth tables for Next State and Output logic

6

  - Use the state transition table and encodings

- Stop Light

Current State S	Inputs T_{NS}	T_{EW}	Next State S'
ST0	0	x	ST1
ST0	1	x	ST0
ST1	x	x	ST2
ST2	x	0	ST3
ST2	x	1	ST2
ST3	x	x	ST0

Current State S1	S0	Inputs T_{NS}	T_{EW}	Next State S1'	S0'
0	0	0	x	0	1
0	0	1	x	0	0
0	1	x	x	1	0
1	0	x	0	1	1
1	0	x	1	1	0
1	1	x	x	0	0

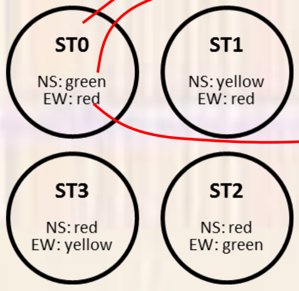
State Machine Design

- Design Process – Create truth tables for Next State and Output logic

6

  - Use the state transition table and encodings

- Stop Light



Output	Encoding
Green	00
Yellow	01
Red	10

Current State		Outputs			
S1	S0	$L_{NS}1$	$L_{NS}0$	$L_{EW}1$	$L_{EW}0$
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

State Machine Design

- Design Process – Minimize the Next State and Output logic

7

  - Boolean Algebra or K-maps

- Stop Light

Current State S1	S0	Inputs T_{NS}	T_{EW}	Next State S1’	S0’
0	0	0	x	0	1
0	0	1	x	0	0
0	1	x	x	1	0
1	0	x	0	1	1
1	0	x	1	1	0
1	1	x	x	0	0

$$S1' = \overline{S1}S0 + S1\overline{S0}\,\overline{T_{EW}} + S1\overline{S0}T_{EW}$$
$$= \overline{S1}S0 + S1\overline{S0}$$
$$= S1 \oplus S0$$

$$S0' = \overline{S1}\,\overline{S0}\,\overline{T_{NS}} + S1\overline{S0}\,\overline{T_{EW}}$$

State Machine Design

- Design Process – Minimize the Next State and Output logic

7

- Boolean Algebra or K-maps

- Stop Light

Current State		Outputs			
S1	S0	$L_{NS}1$	$L_{NS}0$	$L_{EW}1$	$L_{EW}0$
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

$$L_{NS}1 \quad \begin{array}{c|c|c|} & \multicolumn{2}{c}{S1} \\ \cline{2-3} & \mathbf{0} & \mathbf{1} \\ \cline{2-3} S0 & 0 & 1 \\ \cline{2-3} \end{array} \quad S1$$

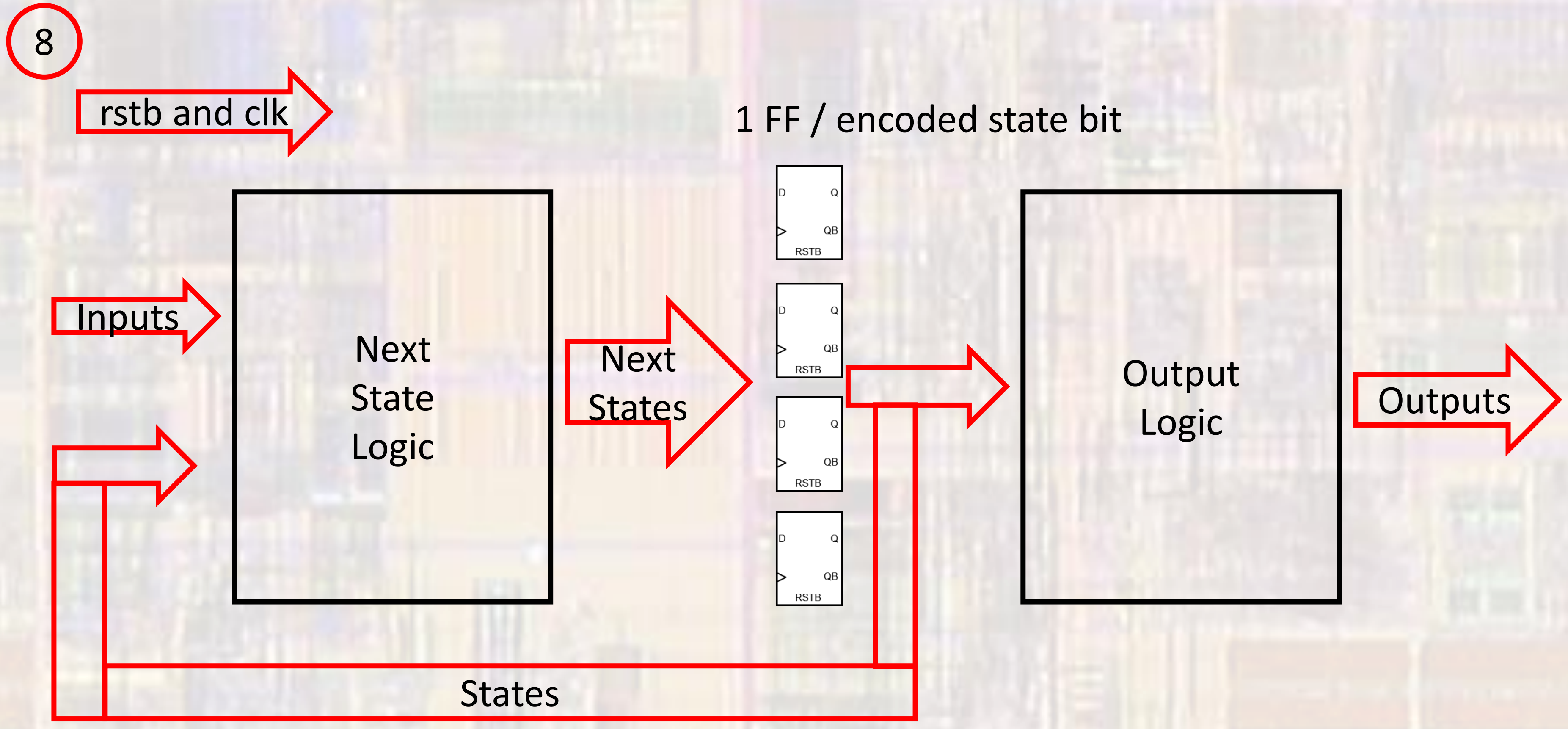
$$L_{NS}0 \quad \begin{array}{c|c|c|} & \multicolumn{2}{c}{S1} \\ \cline{2-3} & \mathbf{0} & \mathbf{0} \\ \cline{2-3} S0 & 1 & 0 \\ \cline{2-3} \end{array} \quad \overline{S1}S0$$

$$L_{EW}1 \quad \begin{array}{c|c|c|} & \multicolumn{2}{c}{S1} \\ \cline{2-3} & \mathbf{1} & \mathbf{0} \\ \cline{2-3} S0 & 1 & 0 \\ \cline{2-3} \end{array} \quad \overline{S1}$$

$$L_{EW}0 \quad \begin{array}{c|c|c|} & \multicolumn{2}{c}{S1} \\ \cline{2-3} & \mathbf{0} & \mathbf{0} \\ \cline{2-3} S0 & 0 & 1 \\ \cline{2-3} \end{array} \quad S1S0$$

State Machine Design

- Schematic

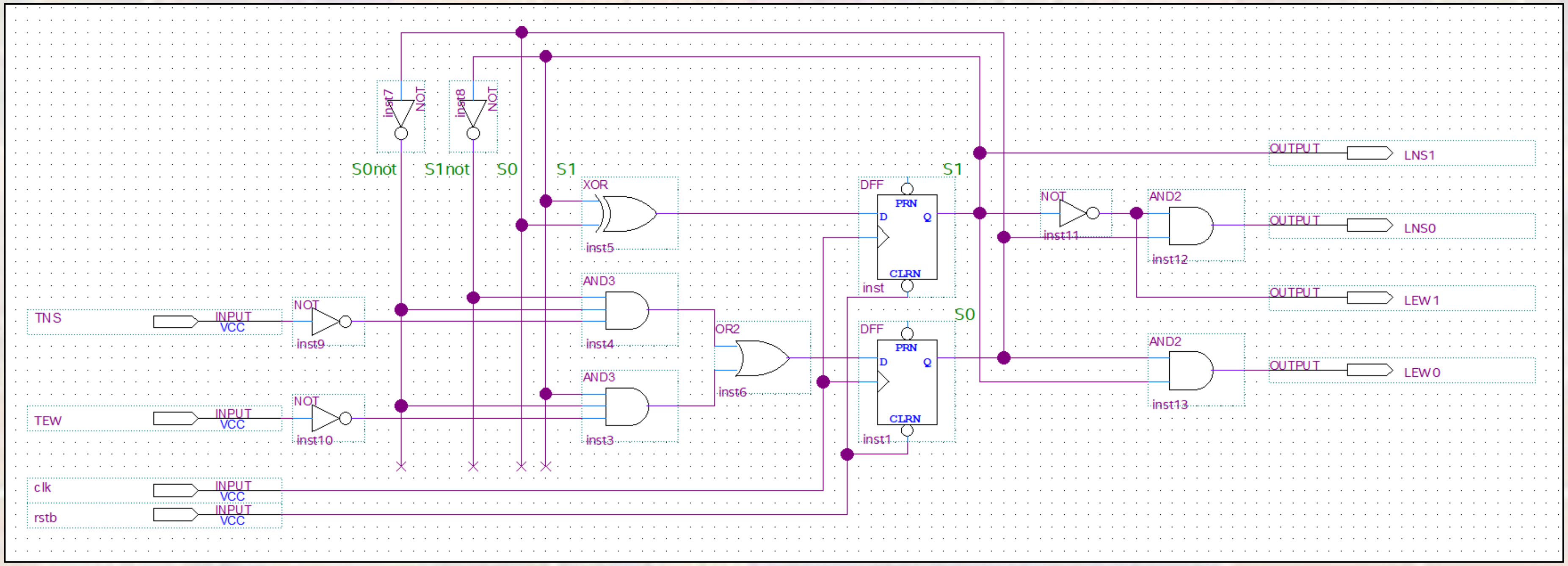


State Machine Design

- Design Process – Draw / create the schematic
  - Quartus

8

- Stop Light



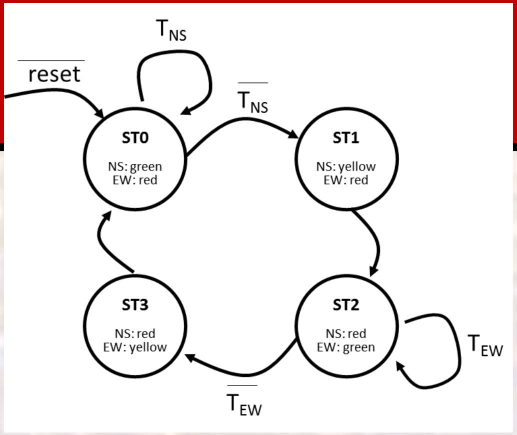
State Machine Design

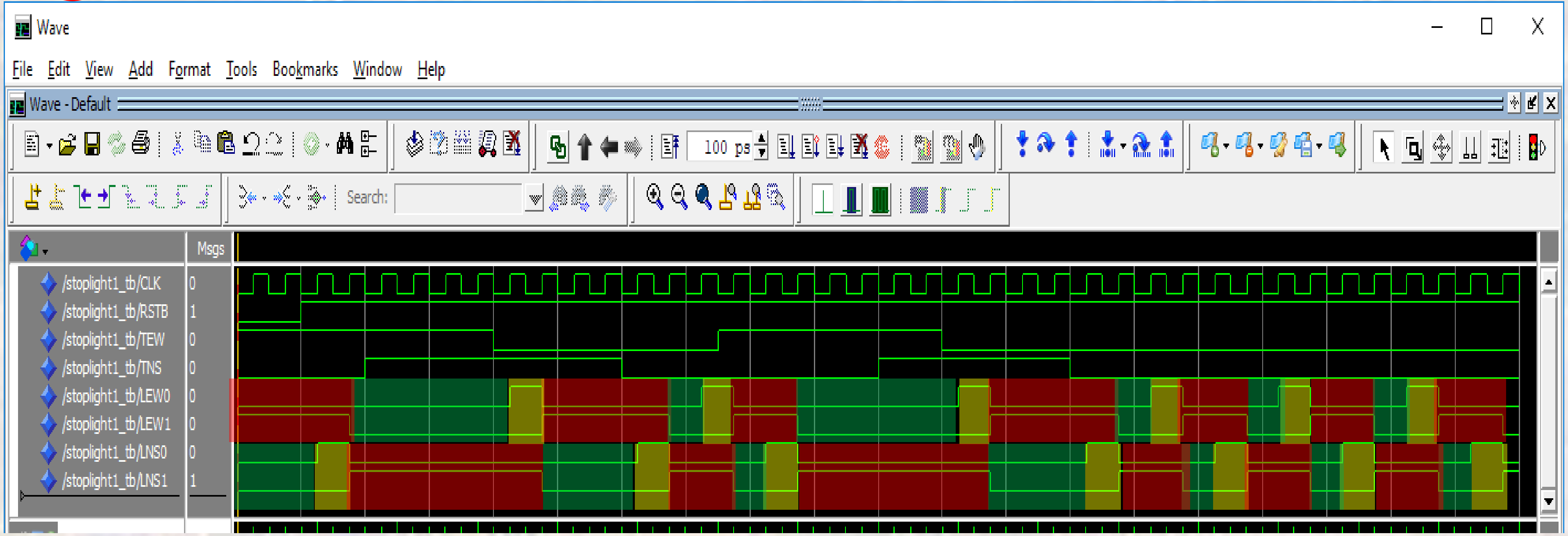
- Verify the solution
  - Design Process
    - Create VHDL file 9
    - Create component file
    - Remove BDF
    - Add VHDL file
    - Create Test bench
    - Elaborate the design
    - Setup simulation
      - Assignments – Settings – simulation
    - Run simulation

State Machine Design

- Verify the solution

9





	Msgs
/stoplight1_tb/CLK	0
/stoplight1_tb/RSTB	1
/stoplight1_tb/TEW	0
/stoplight1_tb/TNS	0
/stoplight1_tb/LEW0	0
/stoplight1_tb/LEW1	0
/stoplight1_tb/LNS0	0
/stoplight1_tb/LNS1	1

Color added by hand