

---

# Lecture 7 FSM Design

7.1 Modeling Finite State Machine

7.2 Mealy and Moore FSMs with Verilog Design

7.3 Design Example: Sequence Detector

7.4 Design Example: Odd/Even Number of 1's Checker

7.5 Design Example: Data Package Receiver

## 7.1 Modeling Finite State Machine

## 7.2 Mealy and Moore FSMs with Verilog Design

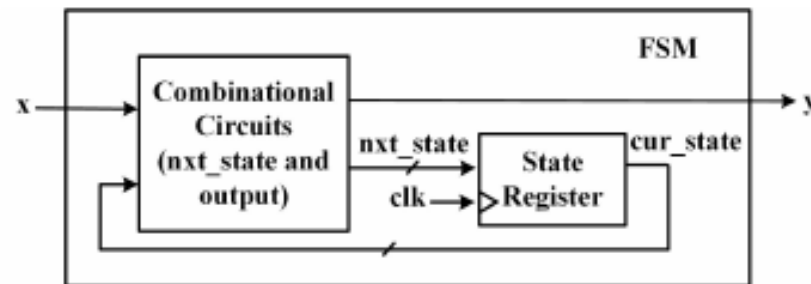
## 7.3 Design Example: Sequence Detector

## 7.4 Design Example: Odd/Even Number of 1's Checker

## 7.5 Design Example: Data Package Receiver

## 7.1.1 Introduction to FSM

- Finite State Machine (FSM)
  - A design component that demonstrates behavior characterized by a finite set of states
    - Combinational circuits: next state and the output
    - Sequential circuit: registers to 1) memorize the status/states; 2) isolate cur\_state from the nxt\_state for timing control
- Template for coding a Finite State Machine (FSM)
  - Sequential circuit: register
  - Combinational circuit of the nxt\_state: always block
  - Combinational circuit of the output: assign/always blocks



**FIGURE 7.1**  
Finite State Machine Model

## 7.1.2 FSM Template with Verilog HDL

- State register
  - Encoding: 4 states so 2-bit width of the parameter SIZE
  - One-hot encoding: S0=4'b0001, S1=4'b0010, S2=4'b0100, and S3=4'b1000.
    - Advantage: Only one state bit ON at a time.
    - Downside: Takes more flip flops.

```
1  module FSM_template (input      rst      ,
2                        input      clk      ,
3                        input      x        ,
4                        output reg y        );
5  parameter SIZE = 2;
6  parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3=2'b11;
7
8  reg [SIZE-1:0] cur_state; // Sequential part of the FSM
9  reg [SIZE-1:0] nxt_state; // Combinational part of the FSM
10
11 //-----State Register -----
12 always @ (posedge clk, negedge rst) begin
13     if (~rst) begin
14         cur_state <= S0;
15     end else begin
16         cur_state <= nxt_state;
17     end
18 end
```

## 7.1.2 FSM Template with Verilog HDL

- Combinational circuit of the next state
- Combinational circuit of the output y
  - The output design could be simplified using *assign* blocks. In such a case, the output signal ``y" must be declared as a *wire* data type.

```
20 //---- Combinational Circuit for the Next State ---
21 always @ (cur_state, x, rst) begin
22     if (~rst) begin
23         nxt_state <= S0;
24     end else begin
25         case(cur_state)
26             S0      :
27             S1      :
28             S2      :
29             S3      :
30             default: nxt_state <= S0;
31         endcase
32     end
33 end
34
```

```
35 //-----Output Combinational Circuit-----
36 always @ (cur_state, x, rst) begin
37     if (~rst) begin
38         y <= 1'b0;
39     end else begin
40         case(cur_state)
41             S0      :
42             S1      :
43             S2      :
44             S3      :
45             default: y <= 1'b0;
46         endcase
47     end
48 end
49 endmodule
```

7.1 Modeling Finite State Machine

7.2 Mealy and Moore FSMs with Verilog Design

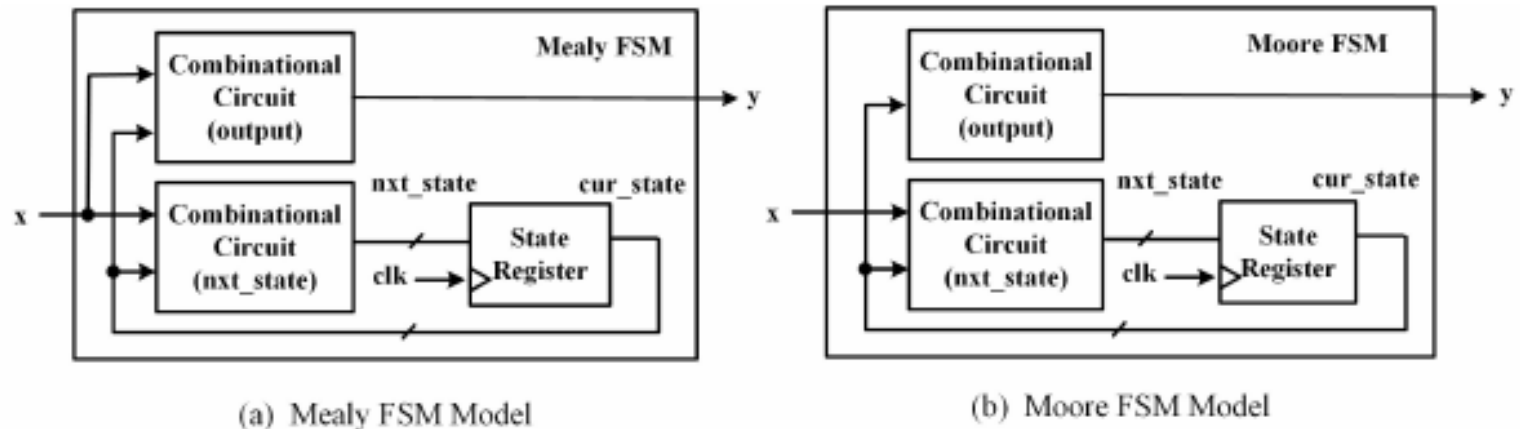
7.3 Design Example: Sequence Detector

7.4 Design Example: Odd/Even Number of 1's Checker

7.5 Design Example: Data Package Receiver

## 7.2.1 Introduction to Mealy and Moore Machine

- Mealy machine:
  - The outputs are associated with both the current state and the inputs during the transition.
- Moore machine:
  - The outputs are solely dependent on the current state and are not influenced by the inputs.



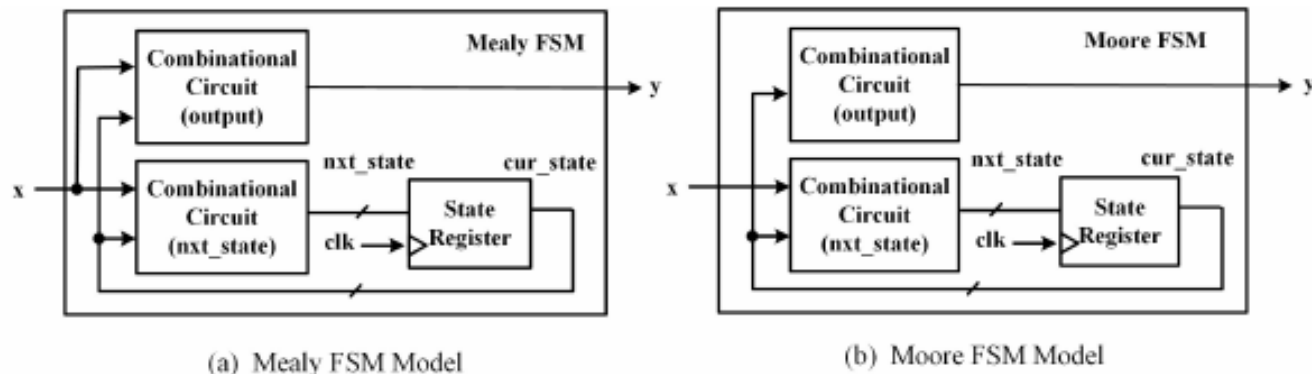
**FIGURE 7.2**

Mealy and Moore FSM Models



## 7.2.1 Introduction to Mealy and Moore Machine

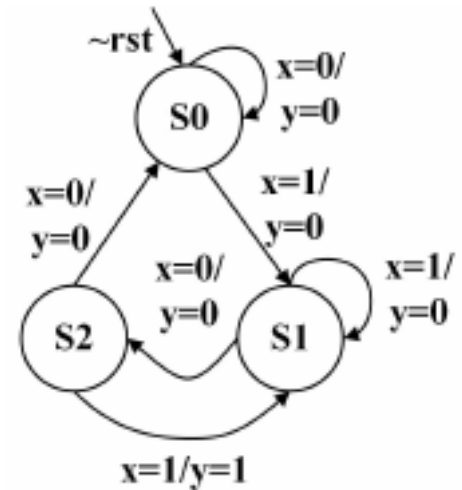
- Designing and evaluating an FSM
  - 1) Constructing a state graph
  - 2) Creating the state table and encoding it into a transition table
    - Synthesis tools in the IC design process can automate the K-map optimization, thus removing the necessity for manual K-map optimization by RTL designers)
  - 3) Designing the FSM using Verilog HDL
  - 4) Analyzing the simulation and synthesis results using Karnaugh maps (commonly known as K-maps).



**FIGURE 7.2**  
Mealy and Moore FSM Models

## 7.2.2.1 State Graph of Mealy Machine

- State graph
  - Three current states, represented by circles labeled as ``S0'', ``S1'', and ``S2''.
    - When the circuit is reset, the state machine initializes to the initial state, ``S0''.
  - The arcs connecting the current states to the next states represent the state transitions that occur over clock cycles.
    - In every clock cycle, one of the arcs will be executed to transition the machine state into the next state.
  - Each arc also includes the input ``x'' and its corresponding output ``y''.
    - The output ``y'' is determined based on the current states (in the circle) and the input ``x'' (on the arc).



**FIGURE 7.3**  
State Graph of Mealy FSM

## 7.2.2.2 State and Transition Tables of Mealy Machine

- State table and Transition table
  - The transition table is presented by encoding the state table.
  - As there are three states being represented, a 2-bit register is required.
    - The ``q1" and ``q0" bits refer to the Most Significant Bit (MSB) and Least Significant Bit (LSB) of the current state, respectively. On the other hand, ``q1+" and ``q0+" represent the MSB and LSB of the next state.
    - The next state (``q1+q0+") serves as the input to the state register, while the current state (``q1q0") serves as the output of the state register.

cur_state	nxt_state		y	
	x=0	x=1	x=0	x=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1

(a) State Table of the Mealy FSM

q1q0	q1+q0+		y	
	x=0	x=1	x=0	x=1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

S0=2'b00, S1=2'b01, S2=2'b10

(b) Transition Table of the Mealy FSM

**FIGURE 7.4**

State and Transition Tables of Mealy FSM

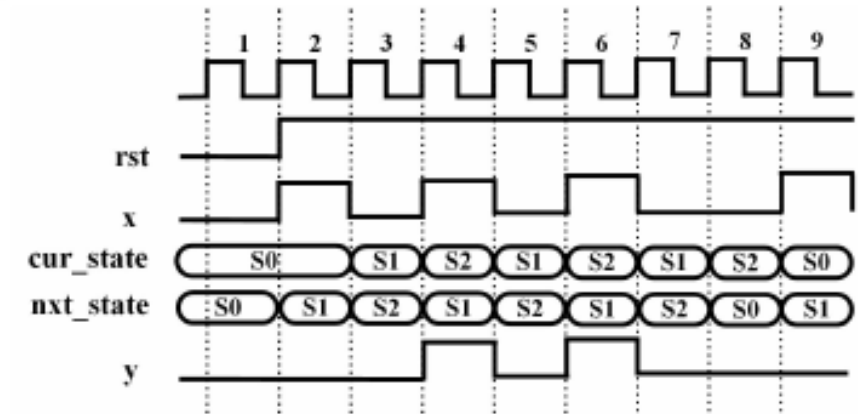
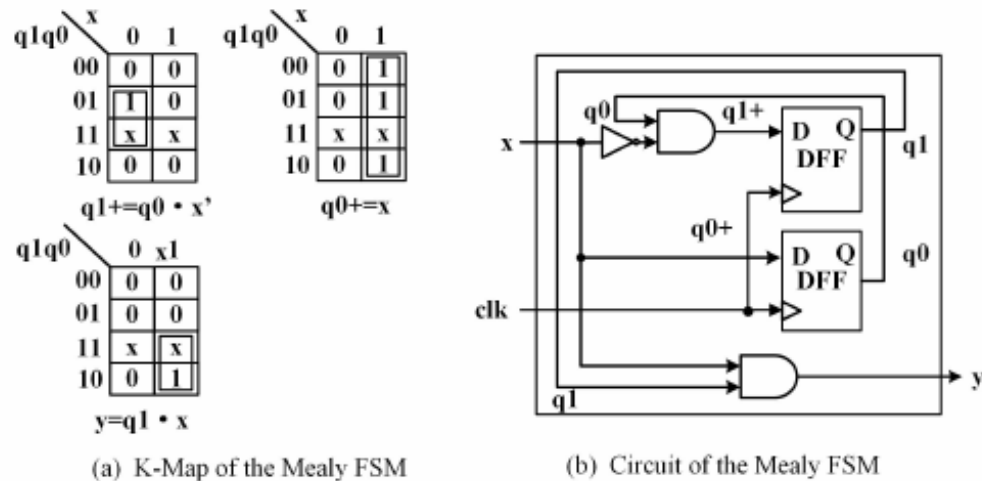
## 7.2.2.3 Verilog HDL Design of Mealy Machine

```
1 module mealy_fsm(input      rst      ,
2                  input      clk      ,
3                  input      x        ,
4                  output     y        );
5
6 parameter SIZE = 2;
7 parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;
8
9 reg [SIZE-1:0] cur_state; // Sequential part of the FSM
10 reg [SIZE-1:0] nxt_state; // Combinational part of the FSM
11
12 //-----State Register -----
13 always @ (posedge clk, negedge rst) begin
14     if (~rst) begin
15         cur_state <= S0;
16     end else begin
17         cur_state <= nxt_state;
18     end
19 end
20
21 //-----Next State Combinational Circuit-----
22 always @ (cur_state, x, rst) begin
23     if (~rst) begin
24         nxt_state = S0;
25     end else begin
26         case(cur_state)
27             S0      : if(x)  nxt_state=S1;
28             S1      : if(~x)  nxt_state=S2;
29             S2      : if(x)  nxt_state=S1; else nxt_state=S0;
30             default: nxt_state = S0;
31         endcase
32     end
33 end
34
35 //-----Output Combinational Circuit-----
36 assign y = (cur_state==S2) & x;
37 endmodule
```

- Verilog code to describe the mealy machine
  - The missing *else* in each case statement represents the scenario where the next state stays the same as the current state..
  - The computation of the output ``y" is described using a concurrent *assign* block in lines 35-36. Therefore, the output ``y" is declared in line 4 in a default *wire* data type.

## 7.2.2.4 Simulation and Synthesis Analysis of Mealy Machine

- A. Synthesized Circuit
- B. Timing Diagram



**FIGURE 7.6**

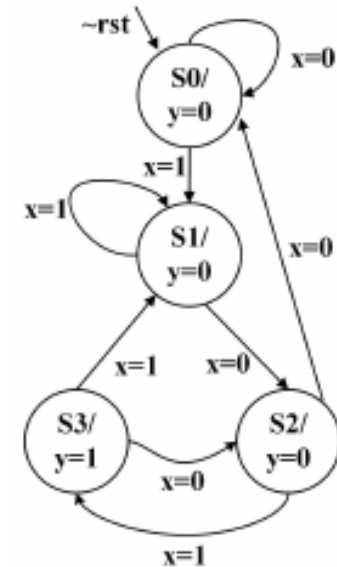
Timing Diagram of Mealy FSM

**FIGURE 7.5**

K-Map Optimization and Circuit of Mealy FSM

## 7.2.3.1 State Graph of Moore Machine

- State graph
  - The current states ("S0", "S1", "S2", and "S3") and the output "y" are all represented within the four circles.
    - Because in a Moore machine the output "y" depends solely on the current state and remains constant within each state.
    - Consequently, it takes an additional state, "S3", to represent the output "y=1" compared to a Mealy machine.
  - The arcs connecting the current states to the next states represent the transitions that happen over clock cycles.



**FIGURE 7.7**  
State Graph of Moore FSM

## 7.2.3.2 State and Transition Tables of Moore Machine

- State table
  - The output ``y" is solely determined by the current state. Therefore, the last column displays the output ``y" without considering the input ``x".
- Transition table
  - The transition of the next state (``q1+q0+") is determined by the combination of the current state (``q1q0") and the input ``x".

cur_state	nxt_state		y
	x=0	x=1	
S0	S0	S1	0
S1	S2	S1	0
S2	S0	S3	0
S3	S2	S1	1

q1q0	q1+q0+		y
	x=0	x=1	
00	00	01	0
01	10	01	0
10	00	11	0
11	10	01	1

S0=2'b00, S1=2'b01,  
S2=2'b10, S3=2'b11

(a) State Table of the Moore FSM      (b) Transition Table of the Moore FSM

**FIGURE 7.8**

State and Transition Tables of Moore FSM

## 7.2.3.3 Verilog HDL Design of Moore Machine

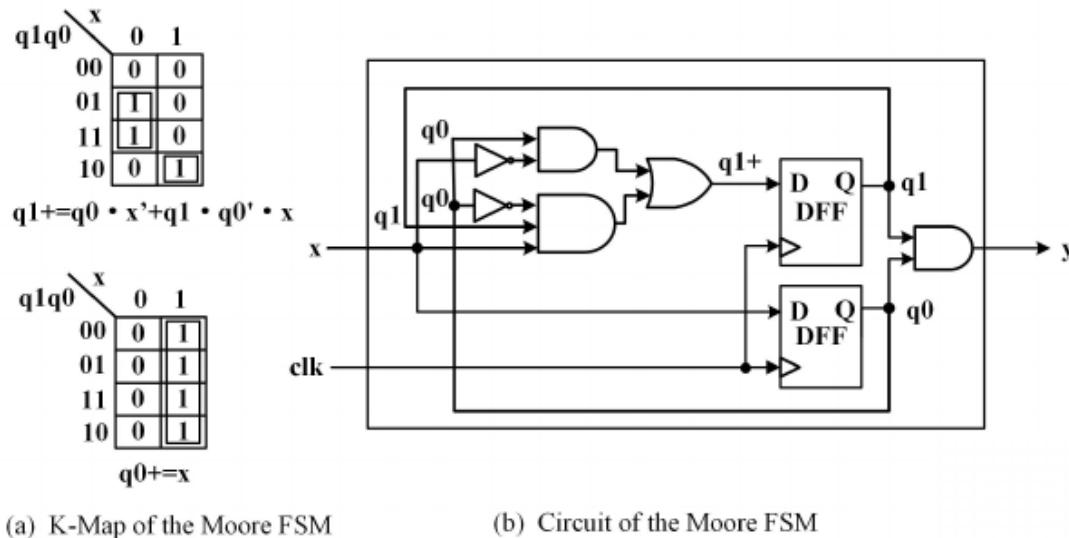
```
1 module moore_fsm(input      rst      ,
2                  input      clk      ,
3                  input      x        ,
4                  output     y        );
5
6 parameter SIZE = 2;
7 parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
8
9 reg [SIZE-1:0] cur_state; // Sequential part of the FSM
10 reg [SIZE-1:0] nxt_state; // Combinational part of the FSM
11
12 //-----State Register -----
13 always @ (posedge clk, negedge rst) begin
14     if (~rst) begin
15         cur_state <= S0;
16     end else begin
17         cur_state <= nxt_state;
18     end
19 end
20
21 //-----Next State Combinational Circuit-----
22 always @ (cur_state, x, rst) begin
23     if (~rst) begin
24         nxt_state = S0;
25     end else begin
26         case(cur_state)
27             S0    : if(x)  nxt_state=S1;
28             S1    : if(~x)  nxt_state=S2;
29             S2    : if(x)  nxt_state=S3; else nxt_state=S0;
30             S3    : if(x)  nxt_state=S1; else nxt_state=S2;
31             default: nxt_state = S0;
32         endcase
33     end
34 end
35
36 //-----Output Combinational Circuit-----
37 assign y = (cur_state==S3);
38 endmodule
```

- Verilog code to describe the mealy machine
  - The output ``y" is set to binary one only when the current state is ``S3". The output computation does not depend on the input ``x".

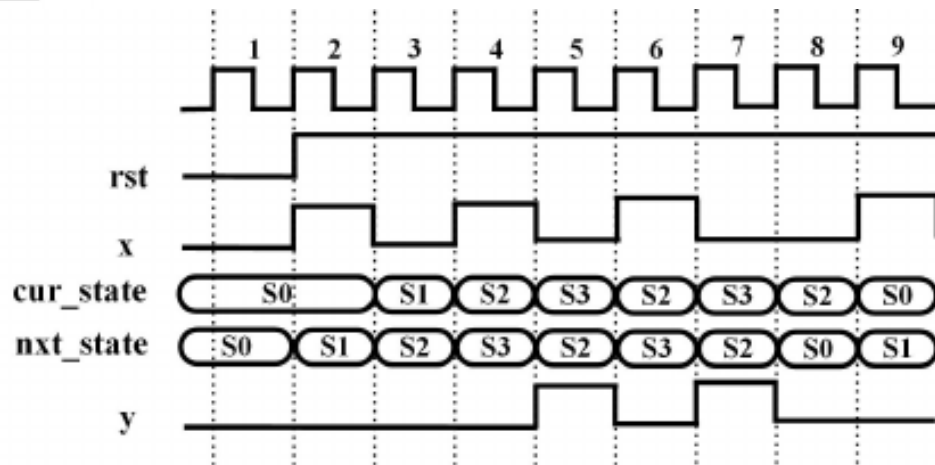


## 7.2.2 Simulation and Synthesis Analysis of Moore Machine

- A. Synthesized Circuit
- B. Timing Diagram



**FIGURE 7.9**  
K-Map Optimization and Circuit of Moore FSM



**FIGURE 7.10**  
Timing Diagram of Moore FSM

7.1 Modeling Finite State Machine

7.2 Mealy and Moore FSMs with Verilog Design

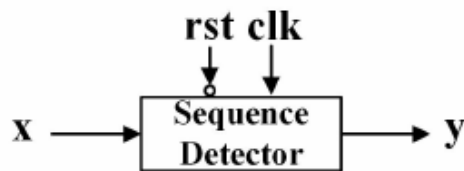
**7.3 Design Example: Sequence Detector**

7.4 Design Example: Odd/Even Number of 1's Checker

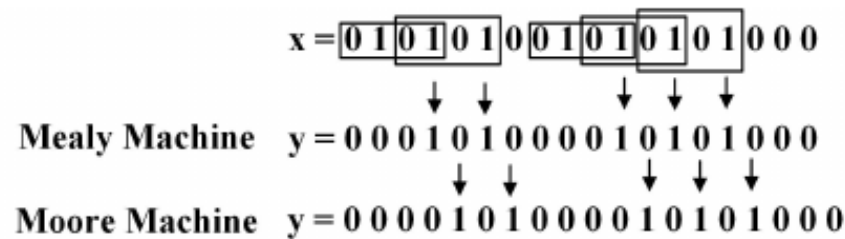
7.5 Design Example: Data Package Receiver

## 7.3.1 Introduction to Sequence Detector

- Introduction to Sequence Detector
  - Asserts an output signal `y=1` when a specified pattern of consecutive bits is received in the serial input stream `x`.
  - The input stream is fed into the detector from left to right, with only one bit digit per clock cycle.



**FIGURE 7.11**  
A Sequence Detector



**FIGURE 7.12**  
Digit String of 0101 Sequence Detector with Mealy and Moore FSM

- The digits in the input can be overlapping between two consecutive occurrences of the desired 0101 string.

## 7.3.1 Introduction to Sequence Detector

---

- Introduction to Sequence Detector
  - The asynchronous reset is utilized for initializing the register,
  - The clock is employed to sample each digit input.
  - A state register is a crucial component in this design, responsible for storing and updating the status of the digit string input.
    - It ensures that the state can be updated on every active clock cycle, enabling the sequence detector to effectively track and detect the desired digit pattern.

**TABLE 7.1**

Sequence Detector IOs Description

Name	Direction	Bit Width	Description
clk	Input	1	Clock, rising edge trigger
rst	Input	1	Asynchronous reset, 0 valid
x	Input	1	Digit string input
y	Output	1	Output 1 when 0101 string detected

## 7.3.2.1 Mealy State Graph of 0101 Sequence Detector

- State graph

- S0: initial state

- From initial state, if 0 input remains to S0; if 1 input goes to S1

- S1: the first desired 0 is received

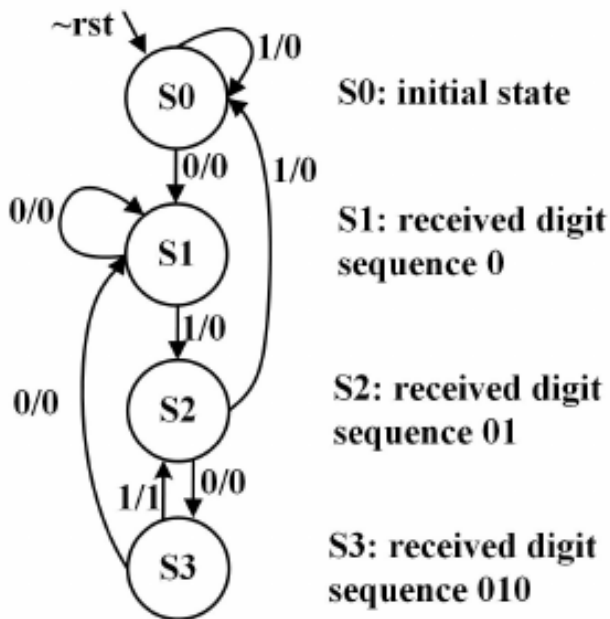
- From S1, if 0 input remains in S1; if 1 input goes to S2

- S2: the two desired sequence 01 is received

- From S2, if 0 input goes to S3; if 1 input goes to S0

- S3: the three desired sequence 010 is received

- From S3, if 0 input goes to S1; if 1 input goes to S2



**FIGURE 7.13**

State Graph of 0101 Sequence Detector Design with Mealy FSM

## 7.3.2.2 Mealy State and Transition Tables of 0101 Sequence Detector

- State table
- Transition table
  - $q1+q0+$ : next state
  - $q1q0$ : current state
  - $y$ : output dependent on the input  $x$  and the current state  $q1q0$

cur_state	nxt_state		y	
	x=0	x=1	x=0	x=1
S0	S1	S0	0	0
S1	S1	S2	0	0
S2	S3	S0	0	0
S3	S1	S2	0	1

q1q0	q1+q0+		y	
	x=0	x=1	x=0	x=1
00	01	00	0	0
01	01	10	0	0
10	11	00	0	0
11	01	10	0	1

$S0=2'b00$ ,  $S1=2'b01$ ,  $S2=2'b10$ ,  $S3=2'b11$

(a) State Table of the 0101 Sequence Detector Design with Mealy FSM

(b) Transition Table of the 0101 Sequence Detector Design with Mealy FSM

**FIGURE 7.14**

State and Transition Tables of 0101 Sequence Detector Design with Mealy FSM

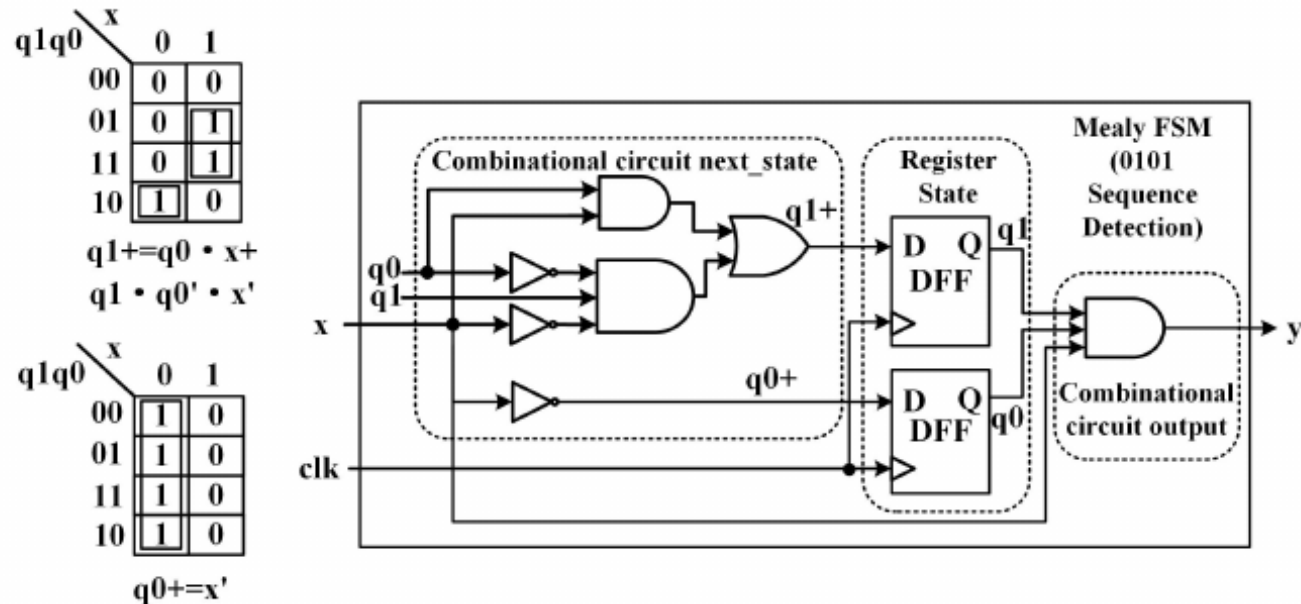
## 7.3.2.3 Verilog HDL Design of 0101 Mealy Sequence Detector

```
1 module seq_det_0101_mealy (input  rst,
2                             input  clk,
3                             input  x ,
4                             output y );
5
6 parameter SIZE = 2;
7 parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
8
9 reg [SIZE-1:0] cur_state; // Sequential part of the FSM
10 reg [SIZE-1:0] nxt_state; // Combinational part of the FSM
11
12 //-----State Register -----
13 always @ (posedge clk, negedge rst) begin
14     if (~rst) begin
15         cur_state <= S0;
16     end else begin
17         cur_state <= nxt_state;
18     end
19 end
20
21 //-----Next state combinational circuit-----
22 always @ (cur_state, x, rst) begin
23     if (~rst) begin
24         nxt_state = 2'b00;
25     end else begin
26         case(cur_state)
27             S0:    if(~x) nxt_state = S1;
28             S1:    if(x)  nxt_state = S2;
29             S2:    if(~x) nxt_state = S3; else nxt_state = S0;
30             S3:    if(~x) nxt_state = S1; else nxt_state = S2;
31             default: nxt_state = S0;
32         endcase
33     end
34 end
35
36 //-----Output combinational circuit-----
37 assign y = (cur_state==S3) & x;
38 endmodule
```

- Verilog code to describe the mealy machine
  - The state register is described in lines 12-19, and the next state transitions is designed in lines 21-34.
  - In line 37, the output signal ``y" is asserted when the current state is ``S3", indicating that the first three digits 010 have been received, and the subsequent input ``x" is one, completing the desired digit string 0101.

## 7.3.2.4 Simulation and Synthesis Analysis of 0101 Mealy Sequence Detector

- A. Synthesized Circuit



(a) K-Map of the 0101 Sequence Detector Design with Mealy FSM

(b) Circuit of the 0101 Sequence Detector Design with Mealy FSM

**FIGURE 7.15**

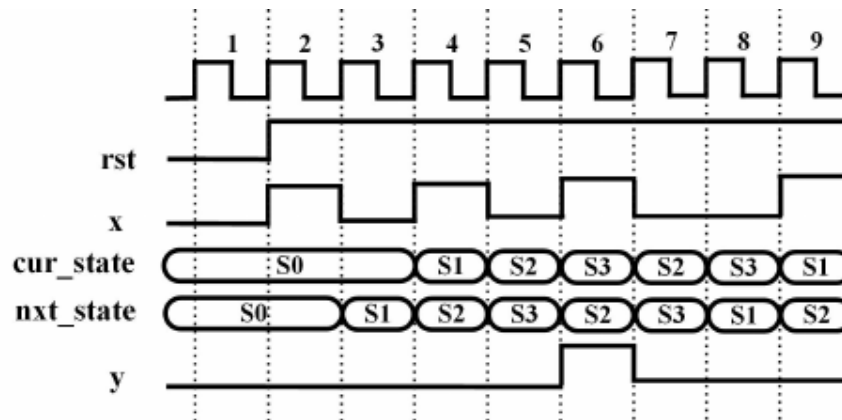
K-Map Optimization and Circuit of 0101 Sequence Detector Design with Mealy FSM



## 7.3.2.4 Simulation and Synthesis Analysis of 0101 Mealy Sequence Detector

### • B. Timing Diagram

- During the second clock cycle, following the reset phase's completion, the current state is set to ``S0''. As the input ``x'' is binary one, which does not match the first desired digit for detecting the string ``0101'', the next state remains ``S0''.
- With the arrival of the next rising clock edge, this ``S0'' state is stored in the register during the subsequent third clock cycle. In this cycle, the detection of the first desired digit zero occurs, leading to a state transition to ``S1'' to signify the reception of the initial desired digit.



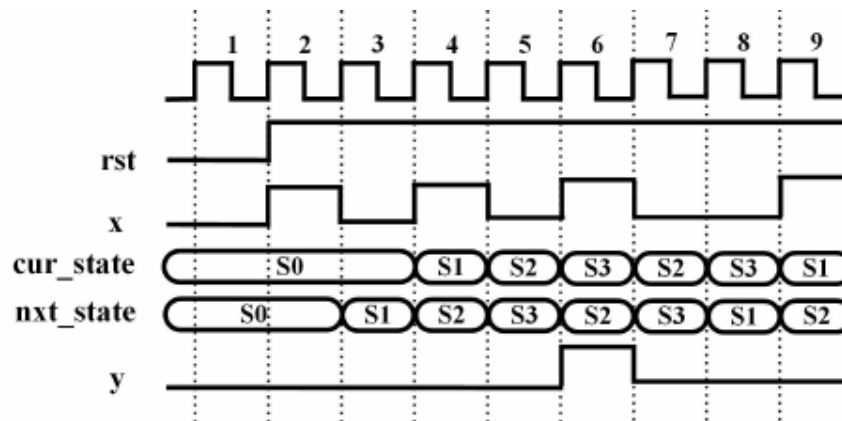
**FIGURE 7.16**

Timing Diagram of the 0101 Sequence Detector Design with Mealy FSM

## 7.3.2.4 Simulation and Synthesis Analysis of 0101 Mealy Sequence Detector

### • B. Timing Diagram

- In clock cycle 4, the current state reflects this transition and shows as ``S1". Since the input is one in this cycle, forming the first two desired string 01, the next state is computed into ``S2".
- In the clock cycle 5, the current state ``S2" and the input zero drives the next state into ``S3". Continuing the state transitions, in clock cycle 6, the current state transitions to ``S3", indicating that three desired digits 010 are received. In the same clock cycle, the ``y" output is asserted because the input ``x" is detected as one within the current ``S3" state, forming the entire string 0101.



**FIGURE 7.16**

Timing Diagram of the 0101 Sequence Detector Design with Mealy FSM

## 7.3.3.1 Moore State Graph of 0101 Sequence Detector

- State graph

- S0: initial state,  $y=0$

- From initial state, if 0 input goes to S1; if 1 input remains to S0

- S1: the 1st desired 0 is received,  $y=0$

- From S1, if 0 input remains in S1; if 1 input goes to S2

- S2: the 2 desired sequence 01 is received,  $y=0$

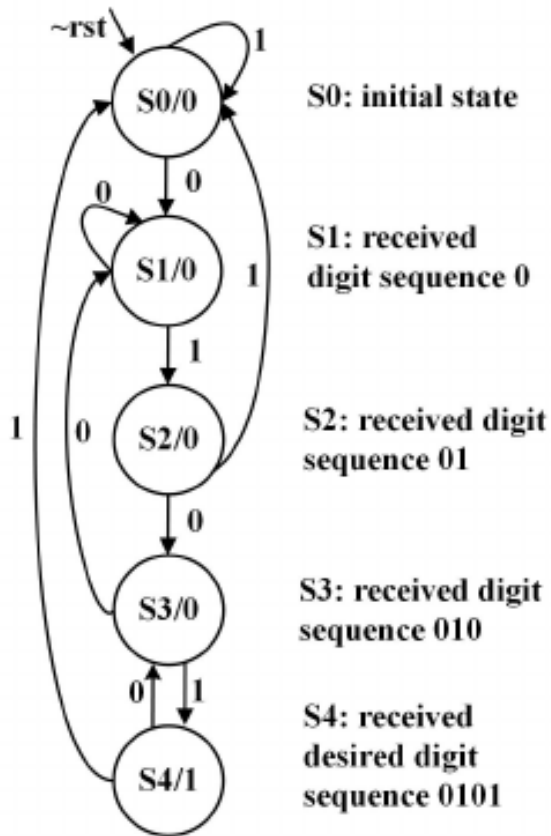
- From S2, if 0 input goes to S3; if 1 input goes to S0

- S3: the 3 desired sequence 010 is received,  $y=0$

- From S3, if 0 input goes to S1; if 1 input goes to S4

- S4: the 4 desired sequence 0101 is received,  $y=0$

- From S4, if 0 input goes to S3; if 1 input goes to S0



**FIGURE 7.17**

State Graph of 0101 Sequence Detector Design with Moore FSM

## 7.3.3.2 Moore State and Transition Tables of 0101 Sequence Detector

- State table
- Transition table
  - In this design, a 3-bit register is required to encode the four states of the Moore machine.
  - Furthermore, the last column of both tables indicates that the output ``y" is solely determined by the current state, regardless of the input ``x".

cur_state	nxt_state		y
	x=0	x=1	
S0	S1	S0	0
S1	S1	S2	0
S2	S3	S0	0
S3	S1	S4	0
S4	S3	S0	1

(a) State Table of the 0101 Sequence Detector Design with Moore FSM

q2q1q0	q2+q1+q0+		y
	x=0	x=1	
000	001	000	0
001	001	010	0
010	011	000	0
011	001	100	0
100	011	000	1

S0=3'b000, S1=3'b001, S2=3'b010,  
S3=3'b011, S4=3'b100

(b) Transition Table of the 0101 Sequence Detector Design with Moore FSM

**FIGURE 7.18**

State and Transition Tables of 0101 Sequence Detector Design with Moore FSM

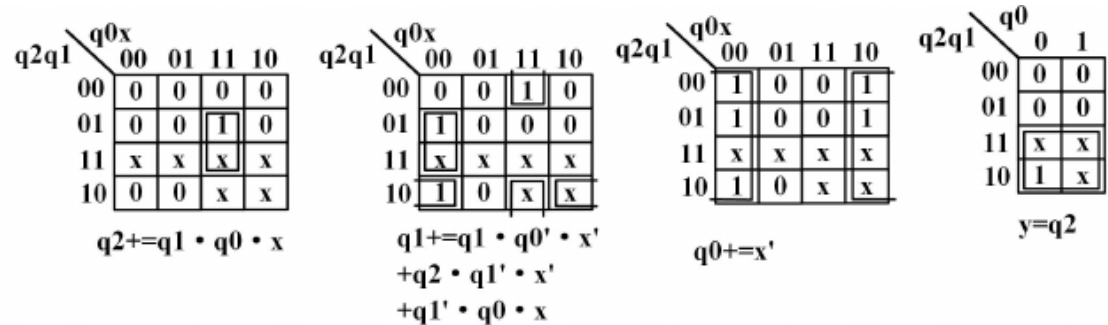
## 7.3.3.3 Verilog HDL Design of 0101 Moore Sequence Detector

```
1  module seq_det_0101_moore (input  rst,
2                             input  clk,
3                             input  x ,
4                             output y  );
5
6  parameter SIZE = 3;
7  parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010,
8             S3 = 3'b011, S4 = 3'b100;
9
10 reg [SIZE-1:0] cur_state; // Sequential part of the FSM
11 reg [SIZE-1:0] nxt_state; // Combinational part of the FSM
12
13 //-----State Register -----
14 always @ (posedge clk, negedge rst) begin
15     if (~rst) begin
16         cur_state <= S0;
17     end else begin
18         cur_state <= nxt_state;
19     end
20 end
21
22 //-----Next state combinational circuit-----
23 always @ (cur_state, x, rst) begin
24     if (~rst) begin
25         nxt_state <= S0;
26     end else begin
27         case(cur_state)
28             S0:    if(~x) nxt_state = S1;
29             S1:    if(x)  nxt_state = S2;
30             S2:    if(x)  nxt_state = S0; else nxt_state = S3;
31             S3:    if(x)  nxt_state = S4; else nxt_state = S1;
32             S4:    if(x)  nxt_state = S0; else nxt_state = S3;
33             default: nxt_state = S0;
34         endcase
35     end
36 end
37
38 //-----Output combinational circuit-----
39 assign y = (cur_state==S4) ;
40 endmodule
```

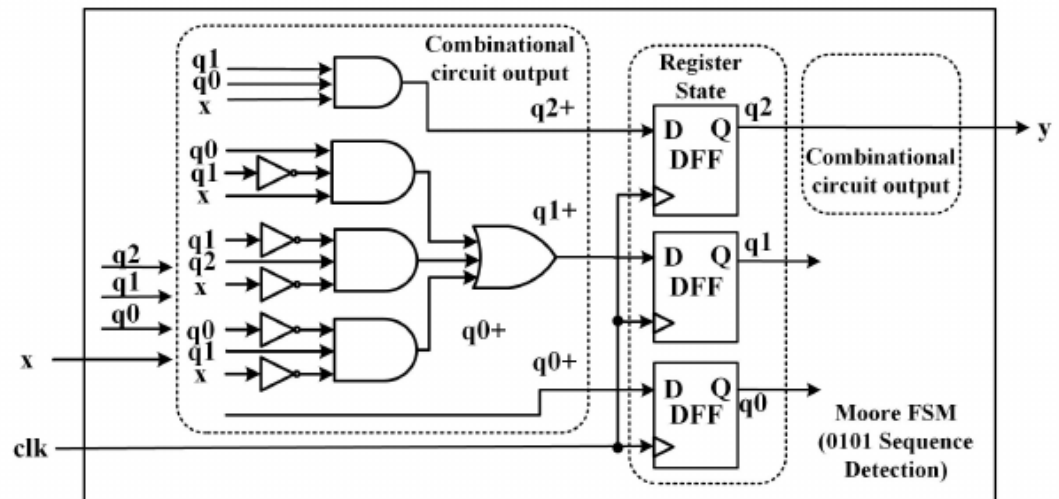
- Verilog code to describe the Moore machine
  - The state register is defined in lines 13-20, while the next state transitions are specified in lines 22-36.
  - In line 39, the output ``y" is activated exclusively when the current state is ``S4". This illustrates the input independence characteristic of a Moore machine.

## 7.3.3.4 Simulation and Synthesis Analysis of 0101 Moore Sequence Detector

- A. Synthesized Circuit



(a) K-Map of the 0101 Sequence Detector Design with Moore FSM



(b) Circuit of the 0101 Sequence Detector Design with Moore FSM

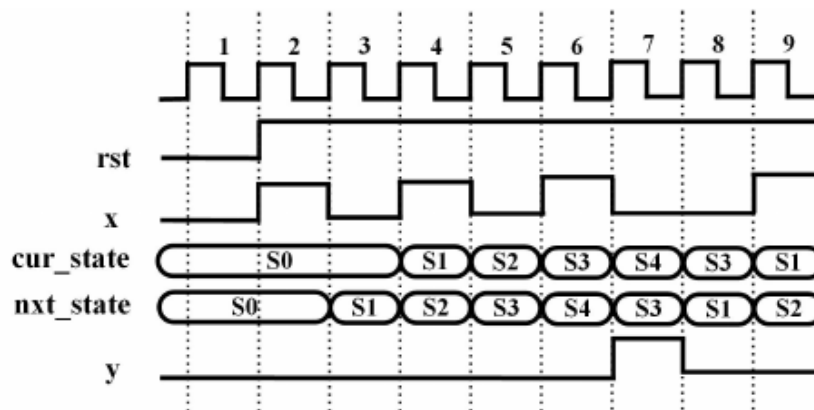
**FIGURE 7.19**

K-Map Optimization and Circuit of 0101 Sequence Detector Design with Moore FSM

## 7.3.3.4 Simulation and Synthesis Analysis of 0101 Moore Sequence Detector

### • B. Timing Diagram

- The sequence detection commences in the second clock cycle, following the reset phase.
- From clock cycles 3 to 5, the input digits ``010" are sequentially received, leading to the machine transitioning to the ``S3" state by clock cycle 6.
- During clock cycle 6, upon receiving the final digit one, the computation of the next state results in ``S4".
- In clock cycle 7, the current state reflects the transition to ``S4", indicating the successful acquisition of the entire digit string ``0101".



**FIGURE 7.20**

Timing Diagram of 0101 Sequence Detector Design with Moore FSM

7.1 Modeling Finite State Machine

7.2 Mealy and Moore FSMs with Verilog Design

7.3 Design Example: Sequence Detector

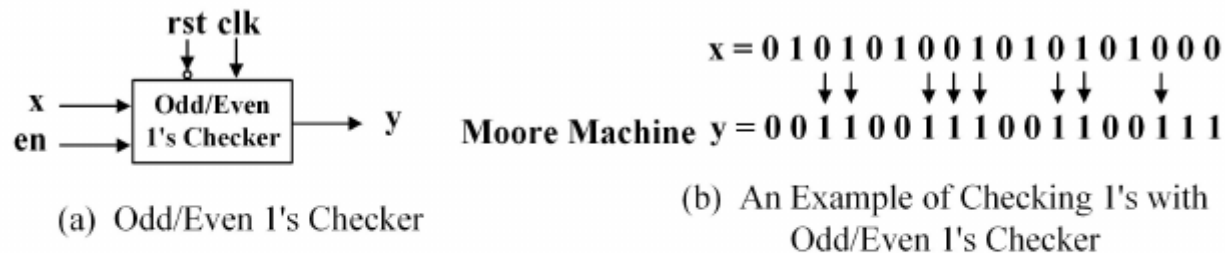
**7.4 Design Example: Odd/Even Number of 1's Checker**

7.5 Design Example: Data Package Receiver



## 7.4.1 Introduction to Odd/Even Number of 1's Checker

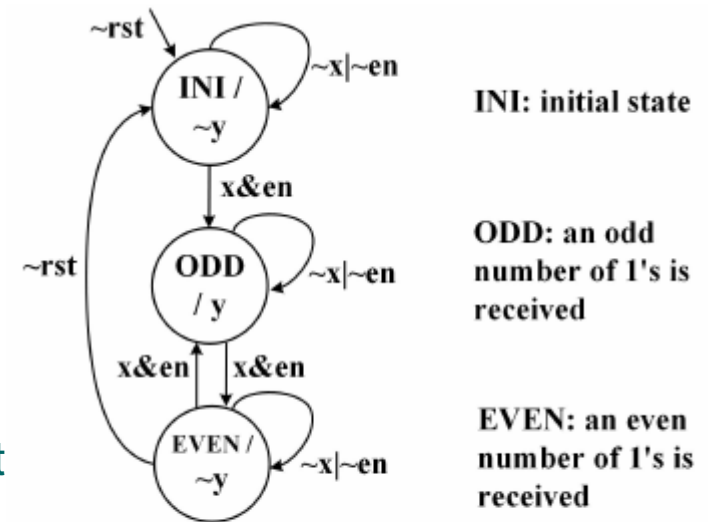
- Odd/Even Number of 1's Checker
  - A digital circuit that determines whether a given binary sequence contains an odd or even number of 1's.
  - This type of circuit is commonly used in
    - Error detection, parity checking, and data transmission.
  - It is a fundamental components in digital circuit that provides important parity information and is essential for ensuring data integrity in various circuits and systems.



**FIGURE 7.21**  
Odd/Even Number of 1's Checker

## 7.4.2 State Graph of Odd/Even 1's Checker

- State Graph of Odd/Even 1's Checker
  - It starts in the initial state ``INI''
  - In the ``ODD'' state,
    - The next state remains in ``ODD'' if the input is zero, as the odd count of 1's continues. If the input is one, the graph transitions to the state ``EVEN'', indicating that an even number of 1's is received.
  - In the ``EVEN'' state,
    - The next state stays in ``EVEN'' if the input is zero, while it transitions back to the ``ODD'' state if the input is one.
  - If the enable signal ``en'' is OFF, the state remains unchanged meaning that the circuit is temporarily inactive.
  - If a reset occurs, the state returns to the initial state ``INI'', since the FSM is initiated.



**FIGURE 7.22**

State Graph of Odd/Even 1's Checker

## 7.4.3 State and Transition Tables of Odd/Even 1's Checker

- State and Transition Tables of Odd/Even 1's Checker
  - Next state transitions:
    - If the circuit is disabled, the state remains unchanged in its current state.
    - As the input zero doesn't alter the ODD or EVEN state, the state also remains identical to the current state
    - The state only updates when both the enable ``en" and ``x" inputs are set to ones.
  - The Boolean sum of all the inputs should evaluate to one, as represented by ``en' + x' + en·x = 1".

cur_state	nxt_state (en, x)			y
	~en	~x	en&x	
INI	INI	INI	ODD	0
ODD	ODD	ODD	EVEN	1
EVEN	EVEN	EVEN	ODD	0

(a) State Table of Odd Even 1's Checker

q1q0	q1+q0+ (en, x)			y
	~en	~x	en&x	
00	00	00	01	0
01	01	01	10	1
10	10	10	01	0

INI=2'b00, ODD=2'b01, EVEN=2'b10

(b) Transition Table of Odd Even 1's Checker

**FIGURE 7.23**

State and Transition Tables of Odd/Even 1's Checker

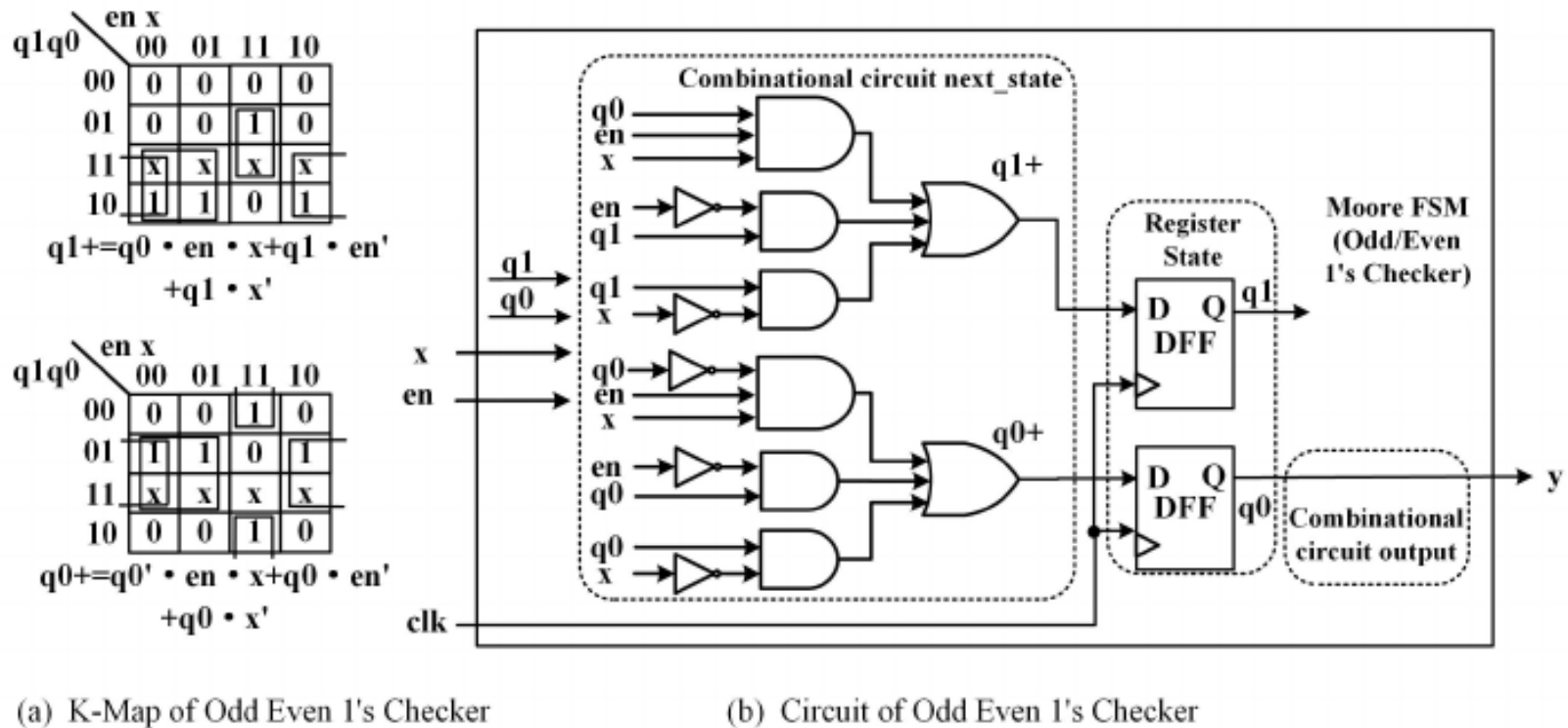
## 7.4.4 Verilog HDL Design of Odd/Even 1's

### Checker

```
1  module odd_even_1s_checker(input  rst  ,
2                             input  clk  ,
3                             input  en   ,
4                             input  x    ,
5                             output y    );
6
7  parameter SIZE = 2;
8  parameter INI  = 2'b00, ODD  = 2'b01, EVEN = 2'b10;
9
10 reg [SIZE-1:0] cur_state; // Sequential part of the FSM
11 reg [SIZE-1:0] nxt_state; // Combinational part of the FSM
12
13 //-----State Register -----
14 always @ (posedge clk or negedge rst) begin
15     if (~rst) begin
16         cur_state <= INI ;
17     end else begin
18         cur_state <= nxt_state;
19     end
20 end
21
22 //-----Next state combinational circuit-----
23 always @ (cur_state, en, x, rst) begin
24     if(~rst) begin
25         nxt_state = INI;
26     end else begin
27         case(cur_state)
28             INI      : if(x&en) nxt_state = ODD ;
29             ODD      : if(x&en) nxt_state = EVEN;
30             EVEN     : if(x&en) nxt_state = ODD ;
31             default  : nxt_state = INI;
32         endcase
33     end
34 end
35
36 //-----Output combinational circuit-----
37 assign y = (cur_state==ODD);
38 endmodule
```

## 7.4.5 Simulation and Synthesis Analysis of Odd/Even 1's Checker

- A. Synthesized Circuit



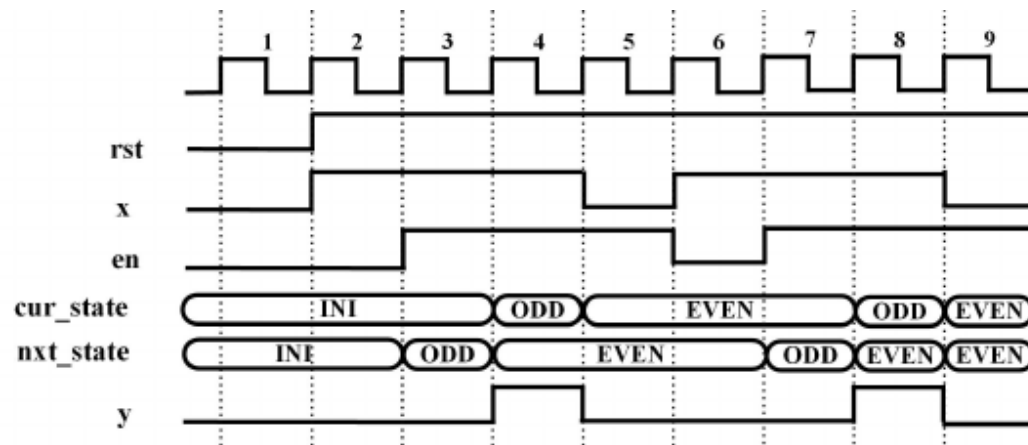
**FIGURE 7.24**

K-Map Optimization and Circuit of Odd/Even 1's Checker

## 7.4.5 Simulation and Synthesis Analysis of Odd/Even 1's Checker

### • B. Timing Diagram

- The entire circuit is reset to the initial states in the first clock cycle. In the second clock cycle, the circuit remains in the initial state ``INI" because the enable signal ``en" is deasserted, indicating that no input is being processed.
- Starting from the third clock cycle, the circuit receives the first digit one as the enable signal is asserted. This triggers a next-state transition from the initial state to the ``ODD" state, indicating that an odd number of 1's is being detected. Due to a Moore machine behavior, consequently, in the fourth clock cycle, the current state is updated into the ``ODD" state and the output ``y" is asserted.



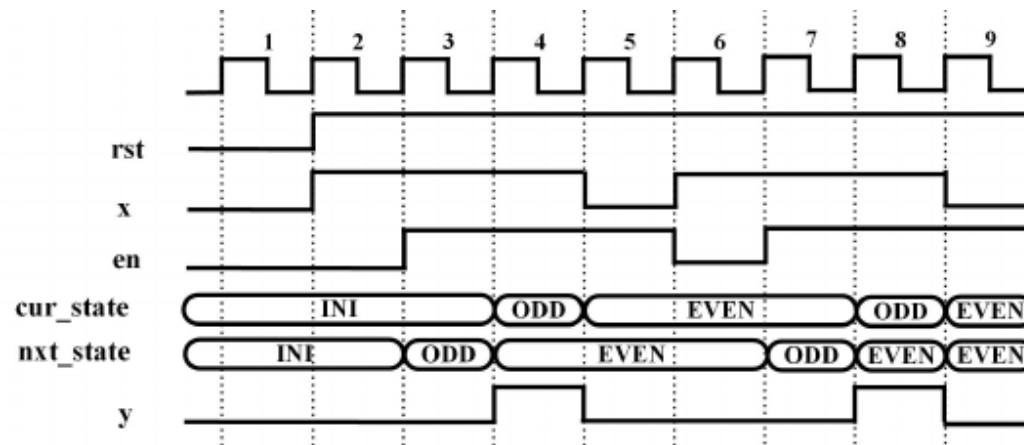
**FIGURE 7.25**

Timing Diagram of Odd/Even 1's Checker

## 7.4.5 Simulation and Synthesis Analysis of Odd/Even 1's Checker

### • B. Timing Diagram

- In clock cycle 4, a continuous digit one is received, leading to the next state transitioning to the ``EVEN'' state. Consequently, in clock cycle 5, the current state reflects the transition to the ``EVEN'' state, causing the output ``y'' to be deasserted.
- The same pattern repeats in subsequent clock cycles, with the circuit transitioning between the ``ODD'' and ``EVEN'' states based on the input sequence of ones and zeros. In clock cycle 8, the output ``y'' is asserted again within the current state of ``ODD''.



**FIGURE 7.25**

Timing Diagram of Odd/Even 1's Checker

7.1 Modeling Finite State Machine

7.2 Mealy and Moore FSMs with Verilog Design

7.3 Design Example: Sequence Detector

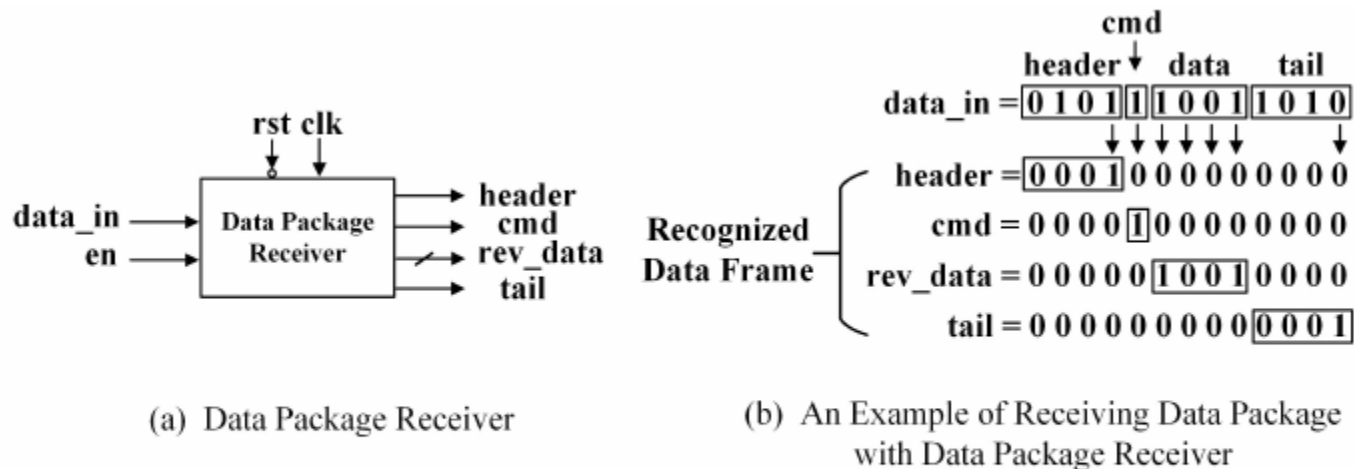
7.4 Design Example: Odd/Even Number of 1's Checker

**7.5 Design Example: Data Package Receiver**



## 7.5.1 Introduction to Data Package Receiver

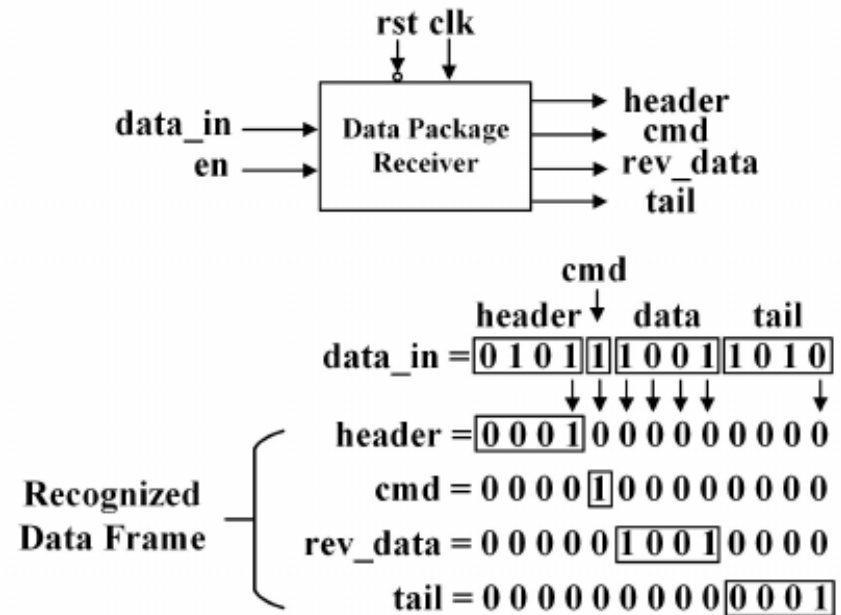
- Introduction to Data Package Receiver
  - In the majority of serial bus communications, data is transmitted in the form of packets, which consist of multiple data segments.
  - Data package header, command, and stop tail through the output indicators ``header'', ``cmd'', and ``tail'', respectively.
  - Furthermore, the extracted data from the data package is shown through the output ``rev\_data''.



**FIGURE 7.26**  
Data Package Receiver

## 7.5.1 Introduction to Data Package Receiver

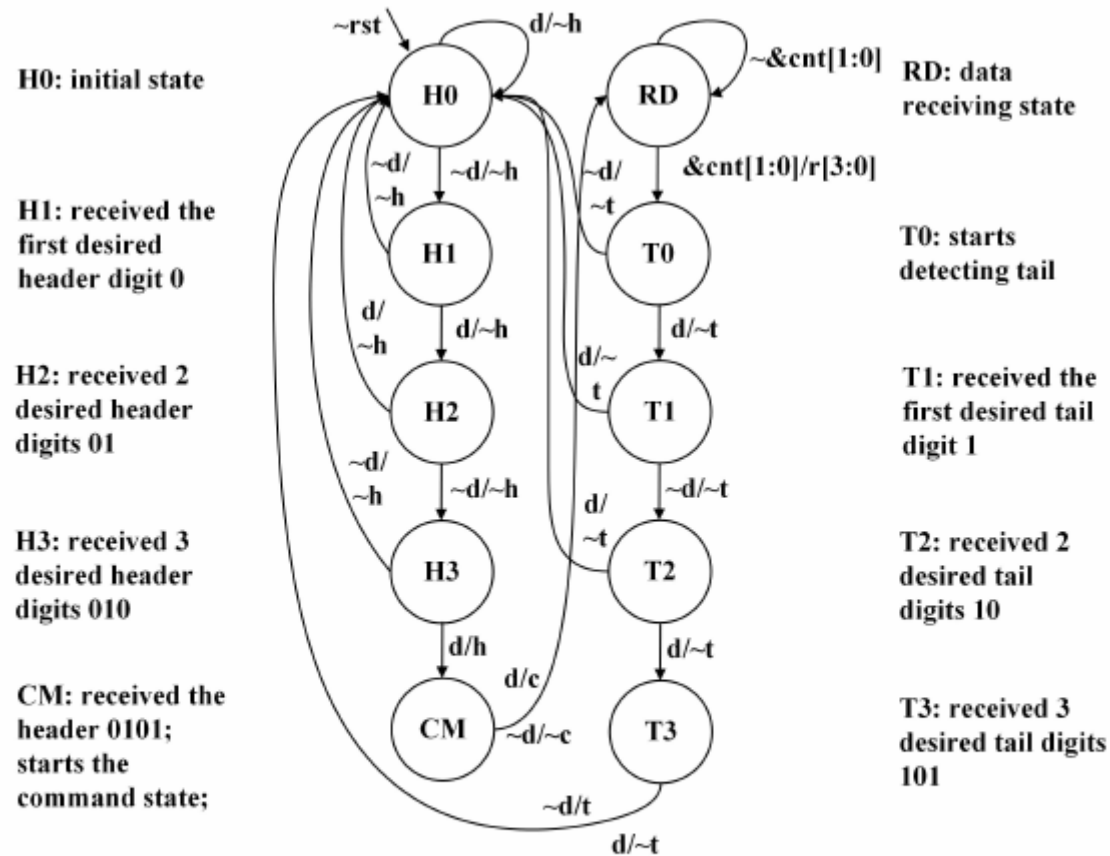
- Data Package Receiver
  - Header: a digit string of 0101 represents the start/head of a data package
  - Command: 1 is a write command and 0 is a read command
  - Received data: copy all 4-bit the data as data received
  - Tail: a digit string of 1010 represents an end/tail of a data package, or a data package is successfully received
- Outputs:
  - Header: header received indicator
  - Cmd: command indicator
  - Rev\_data: received 4-bit data
  - Tail: tail received indicator



**FIGURE 7.26**  
Data Package Receiver

## 7.5.2 State Graph of Data Package Receiver

- State graph
  - H0: initial state
  - H1-H3: header received
  - CM: command state
  - RD: data receiving state
  - T0: starts detecting tail
  - T1-T3: tail received
- Input and output
  - d: data\_in; serial input
  - h: header; indicator output
  - c: cmd; indicator output
  - r: rev\_data; 4-bit output
  - t: tail; indicator output



**FIGURE 7.27**  
State Graph of Data Package Receiver

## 7.5.3 State and Transition Table of Data Package Receiver

cur_state	nxt_state				h		c		t		r[3:0]	
	~d	d	&cnt	~&cnt	~d	d	~d	d	~d	d	&cnt	~&cnt
H0	H1	H0	-	-	0	0	0	0	0	0	-	-
H1	H0	H2	-	-	0	0	0	0	0	0	-	-
H2	H3	H0	-	-	0	0	0	0	0	0	-	-
H3	H0	CM	-	-	0	1	0	0	0	0	-	-
CM	RD	RD	-	-	0	0	0	1	0	0	-	-
RD	-	-	T0	RD	0	0	0	0	0	0	d[3:0]	d
T0	H0	T1	-	-	0	0	0	0	0	0	-	-
T1	T2	H0	-	-	0	0	0	0	0	0	-	-
T2	T3	H0	-	-	0	0	0	0	0	0	-	-
T3	H0	H0	-	-	0	0	0	0	1	0	-	-

(a) State Table of Data Package Receiver

cur_state	nxt_state				h		c		t		r[3:0]	
	~d	d	&cnt	~&cnt	~d	d	~d	d	~d	d	&cnt	~&cnt
0	1	0	-	-	0	0	0	0	0	0	-	-
1	0	2	-	-	0	0	0	0	0	0	-	-
2	3	0	-	-	0	0	0	0	0	0	-	-
3	0	4	-	-	0	1	0	0	0	0	-	-
4	5	5	-	-	0	0	0	1	0	0	-	-
5	-	-	6	5	0	0	0	0	0	0	d[3:0]	d
6	0	7	-	-	0	0	0	0	0	0	-	-
7	8	0	-	-	0	0	0	0	0	0	-	-
8	9	0	-	-	0	0	0	0	0	0	-	-
9	0	0	-	-	0	0	0	0	1	0	-	-

H0=4'd0, H1=4'd1, H2=4'd2, H3=4'd3; CM=4'd4, RD=4'd5;  
T0=4'd6, T1=4'd7, T2=4'd8, T3=4'd9;

(b) Transition Table of Data Package Receiver

- State Table
- Transition Table

**FIGURE 7.28**

State and Transition Tables of Data Package Receiver

## 7.5.4 Verilog HDL Design of Data Package

### Receiver

```
1 module data_pack_rec(input          rst          ,
2                       input          clk          ,
3                       input          en           ,
4                       input          data_in      ,
5                       output         header       ,
6                       output         cmd          ,
7                       output reg [3:0] rev_data   ,
8                       output         tail        );
9
10 parameter SIZE = 4;
11 parameter H0 = 4'b0000;
12 parameter H1 = 4'b0001;
13 parameter H2 = 4'b0010;
14 parameter H3 = 4'b0011;
15 parameter CM = 4'b0100;
16 parameter RD = 4'b0101;
17 parameter T0 = 4'b0110;
18 parameter T1 = 4'b0111;
19 parameter T2 = 4'b1000;
20 parameter T3 = 4'b1001;
21
22 reg [SIZE-1:0] cur_state; // Sequential part of the FSM
23 reg [SIZE-1:0] nxt_state; // Combinational part of the FSM
24
25 //-----State Register -----
26 always @ (posedge clk, negedge rst) begin
27     if(~rst) begin
28         cur_state <= H0 ;
29     end else begin
30         cur_state <= nxt_state;
31     end
32 end
```

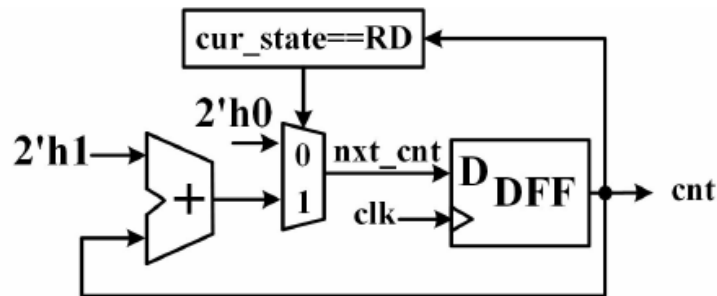
- Sequential part of the FSM

## 7.5.4 Verilog HDL Design of Data Package Receiver

```

33 //-----Next state combinational circuit-----
34 reg [1:0] cnt;
35 wire [1:0] nxt_cnt = (cur_state==RD) ? cnt+2'h1 : 2'h0;
36 always @ (posedge clk, negedge rst) begin
37     if(~rst) begin
38         cnt <= 2'h0 ;
39     end else begin
40         cnt <= nxt_cnt;
41     end
42 end

```



**FIGURE 7.29**  
Data Receiver Counter

- Combinational part of the FSM

```

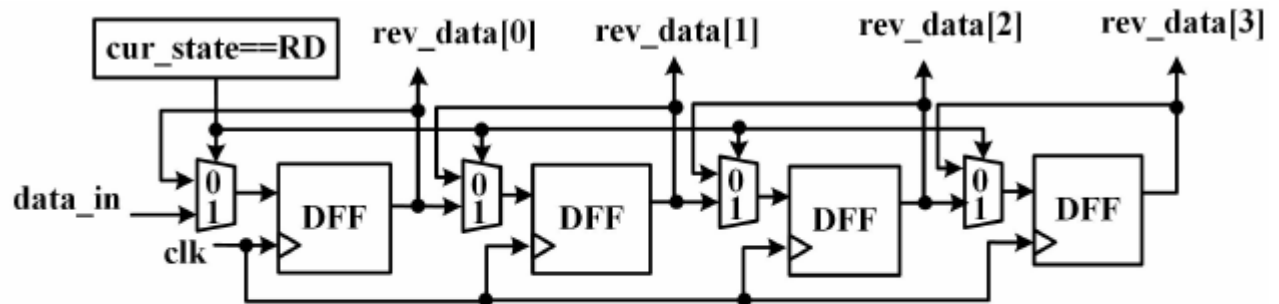
44 always @ (cur_state, en, data_in, rst, cnt) begin
45     if(~rst) begin
46         nxt_state = H0;
47     end else begin
48         case(cur_state)
49             H0      : if(~data_in & en) nxt_state <= H1;
50             H1      : if(data_in )    nxt_state <= H2;
51                     else              nxt_state <= H0;
52             H2      : if(~data_in)    nxt_state <= H3;
53                     else              nxt_state <= H0;
54             H3      : if(data_in )    nxt_state <= CM;
55                     else              nxt_state <= CM;
56             CM      :                nxt_state <= RD;
57             RD      : if(&cnt)         nxt_state <= T0;
58             T0      : if(data_in)     nxt_state <= T1;
59                     else              nxt_state <= H0;
60             T1      : if(~data_in)    nxt_state <= T2;
61                     else              nxt_state <= H0;
62             T2      : if(data_in)     nxt_state <= T3;
63                     else              nxt_state <= H0;
64             T3      :                nxt_state <= H0;
65             default : nxt_state = H0;
66         endcase
67     end
68 end

```

## 7.5.4 Verilog HDL Design of Data Package Receiver

- Output part of the FSM

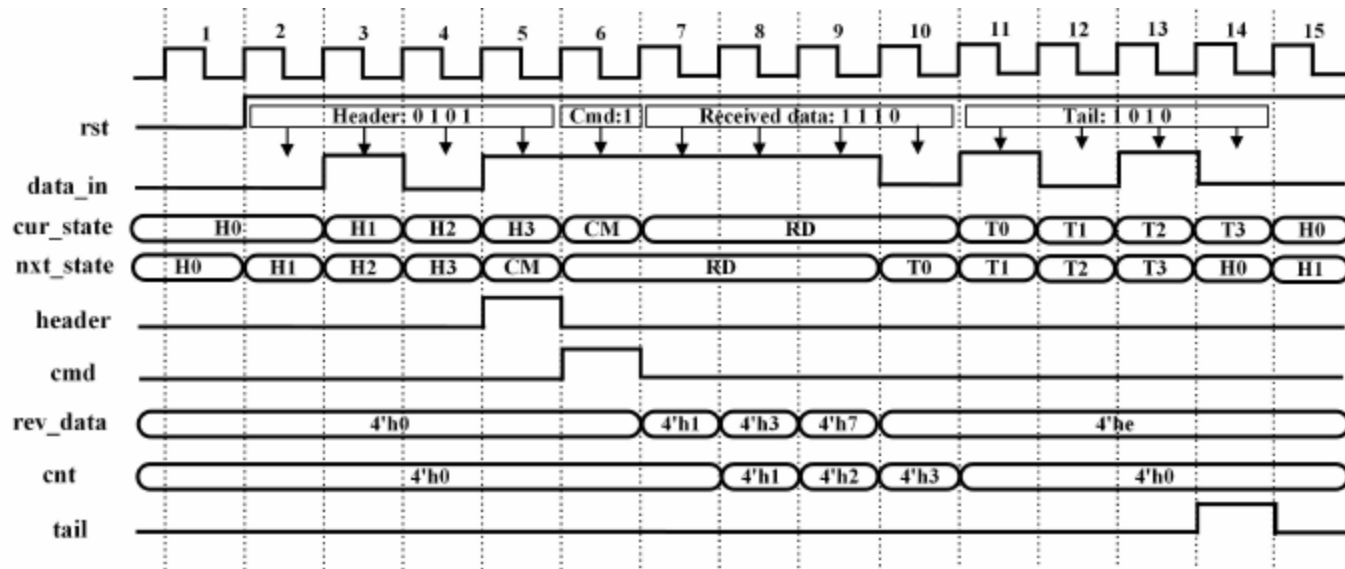
```
70 //-----Output combinational circuit-----
71 assign header = (cur_state==H3) & data_in ;
72 assign cmd     = (cur_state==CM) & data_in ;
73 assign tail    = (cur_state==T3) & ~data_in;
74
75 wire [3:0] nxt_rev_data;
76 assign nxt_rev_data=(cur_state==RD)?{rev_data[2:0],data_in}
77                                     : rev_data;
78 always @ (posedge clk, negedge rst) begin
79     if(~rst) begin
80         rev_data <= 4'h0 ;
81     end else begin
82         rev_data <= nxt_rev_data;
83     end
84 end
85 endmodule
```



**FIGURE 7.30**  
Data Receiver Shifter

## 7.5.4 Simulation and Synthesis Analysis of Data Package Receiver

- Simulation Analysis
  - In the first clock cycle, the entire state machine is reset.
  - From clock cycles 2 to 5, the current state transitions from ``H0" to ``H3".
  - Within the clock cycle 5, the data input becomes logic one, indicating the successful reception of the header string 0101. Consequently, the ``header" indicator activates in the same clock cycle.

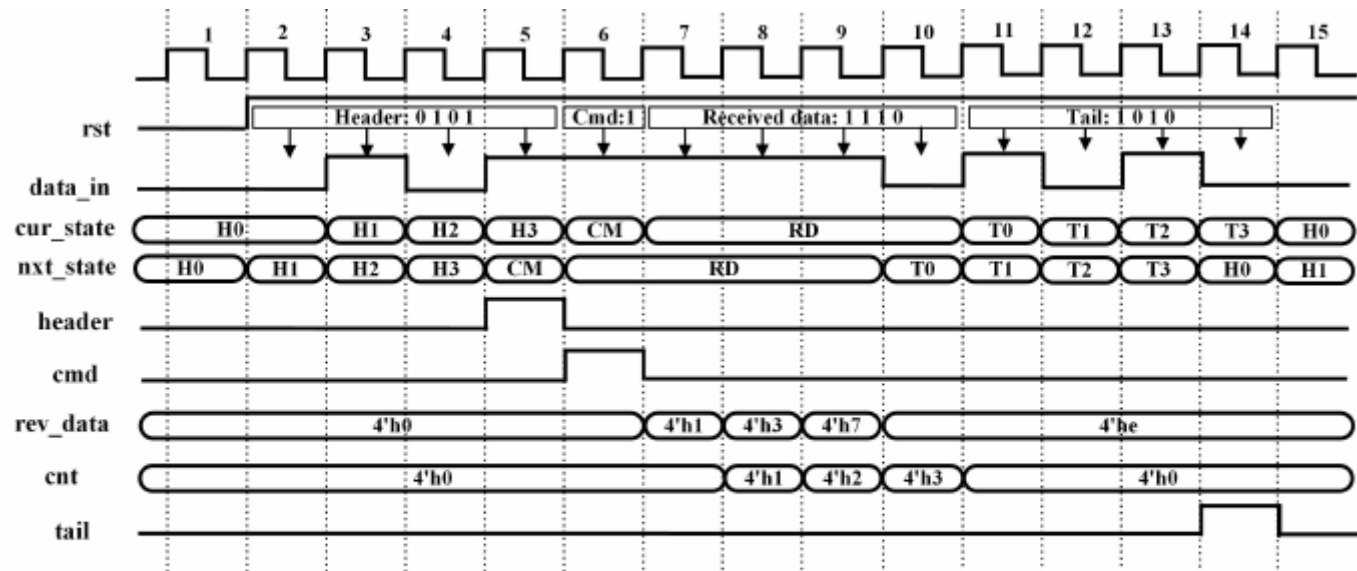


**FIGURE 7.31**  
Timing Diagram of Data Package Receiver



## 7.5.4 Simulation and Synthesis Analysis of Data Package Receiver

- Simulation Analysis
  - Similarly, the current state transitions to ``CM" in clock cycle 6. Since the current input is logic one, the ``cmd" indicator reflects the command as a status data transaction.
  - In the subsequent four clock cycles, 7 to 10, the current state transitions into the ``RD" state, and the digit string 1110 is received and shown on the ``rev\_data[3:0]" output.

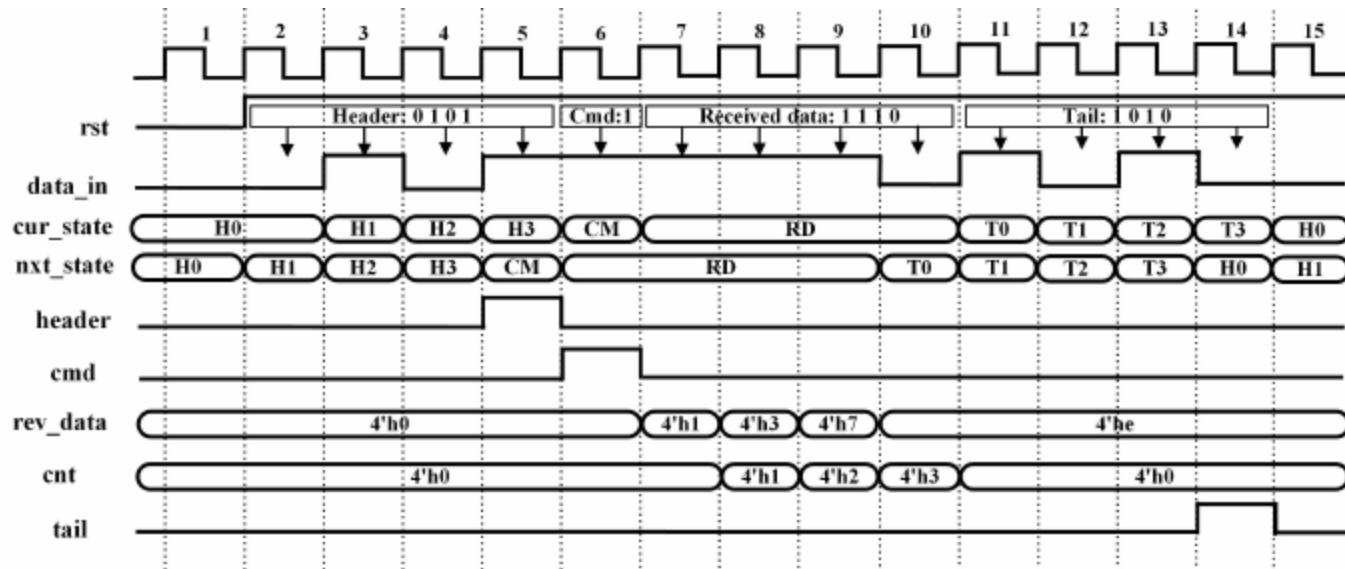


**FIGURE 7.31**

Timing Diagram of Data Package Receiver

## 7.5.4 Simulation and Synthesis Analysis of Data Package Receiver

- Simulation Analysis
  - In the final four clock cycles, 11 to 14, the current state transitions through the ``T0-T3" states to detect the tail in the data package. Once the tail digit string 1010 is successfully extracted, the final ``tail" indicator is asserted, indicating the completion of the data package reception.



**FIGURE 7.31**

Timing Diagram of Data Package Receiver