# DSLab.14  Registers & Counters

# Lab. 14 Registers & Counters

- Design and Verify the following circuits using Verilog HDL
  - Four-bit universal shift register
  - BCD synchronous counter

- Please write and upload the lab report (Lab14) -- Due on 2022/12/09 23:59

## Example 1: State Diagram-Based HDL Model (Moore) (1/4)

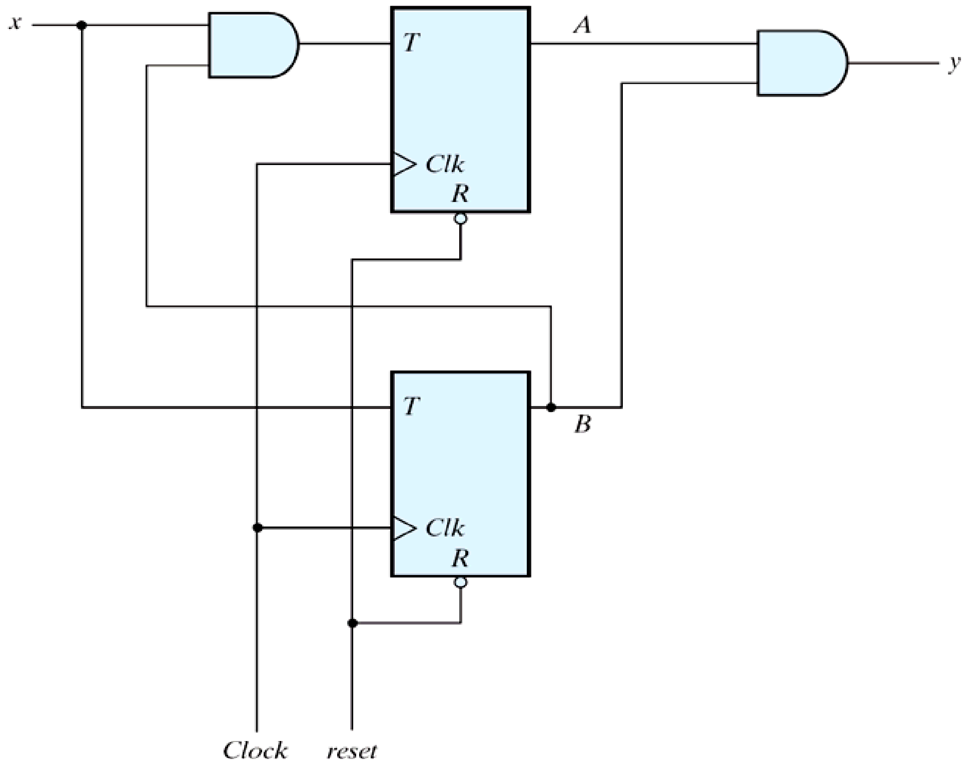
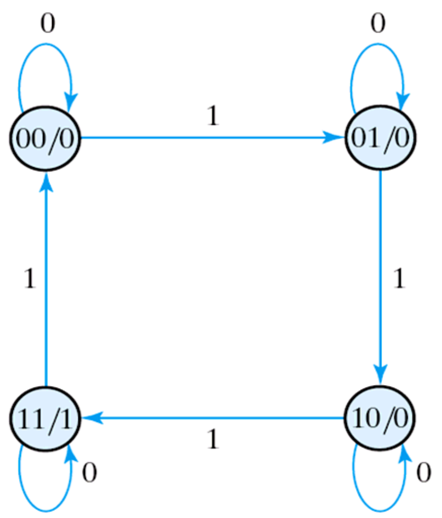


Fig. 20

| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| *A* | *B* | *x* | *A* | *B* | *y* |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |



## Example 1: State Diagram-Based HDL Model (Moore) (2/4)

### Structural model (Fig. 20)

```
module Moore_Model_STR_Fig_5_20 (
   output   y_out, A, B,
   input    x_in, clock, reset
);
  wire      A, TB;

// Flip-flop input equations
    assign TA = x_in & B;
    assign TB = x_in;
//output equation
    assign y_out = A & B;
// Instantiate Toggle flip-flops
    Toggle_flip_flop_3 M_A (A, TA, clock, reset);
    Toggle_flip_flop_3 M_B (B, TB, clock, reset);

endmodule
```

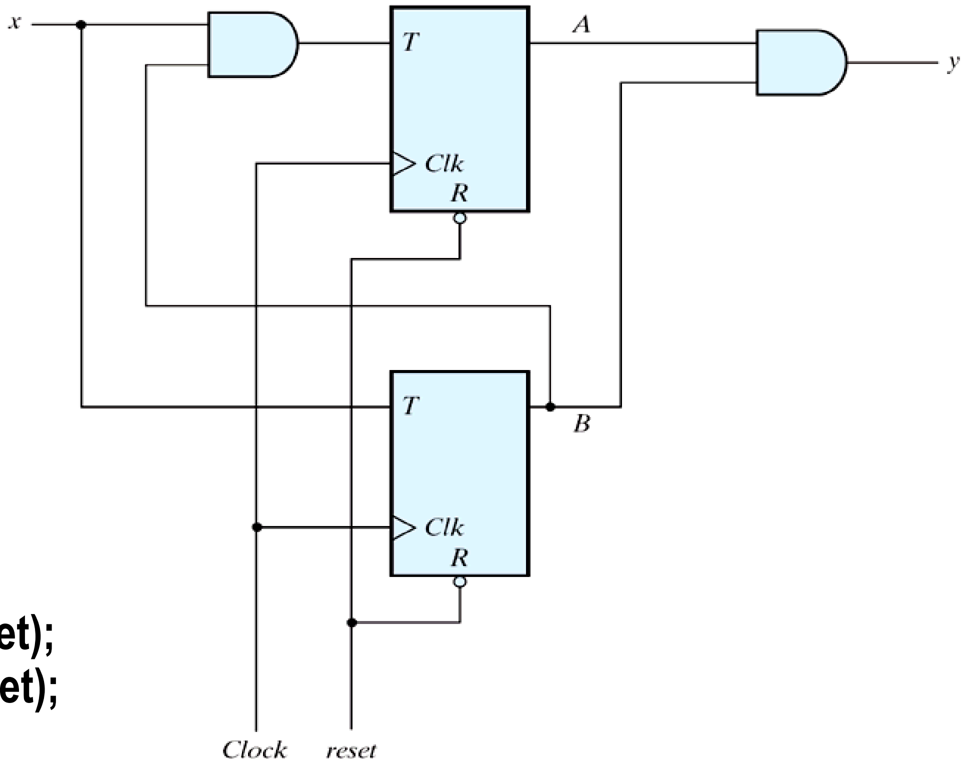


Fig. 20

## Example 1: State Diagram-Based HDL Model (Moore) (3/4)

### Behavioral model (State diagram)

```
module Moore_Model_Fig_5_20 (
   output   y_out,
   input    x_in, clock, reset
);
   reg [1: 0]   state;
   parameter  S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

   always @ (posedge clock, negedge reset)
    if (reset == 0) state <= S0;    // Initialize to state S0
    else case (state)
       S0: if (x_in)  state <= S1; else state <= S0;
       S1: if (x_in)  state <= S2; else state <= S1;
       S2: if (x_in)  state <= S3; else state <= S2;
       S3: if (x_in)  state <= S0; else state <= S3;
    endcase

   assign y_out = (state == S3); // Output of flip-flops
endmodule
```

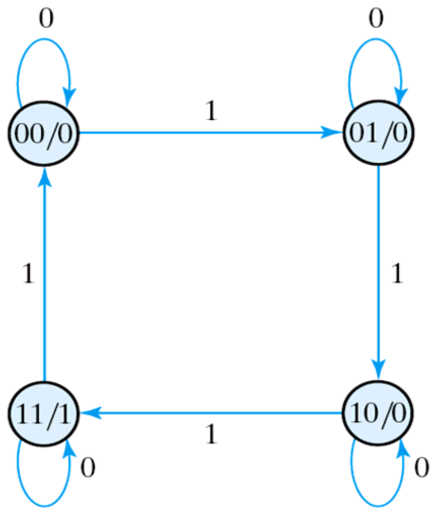


Fig. 5.20

| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| A | B | x | A | B | y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

## Example 1: State Diagram-Based HDL Model (Moore) (4/4)

```
module t_Moore_Fig_5_20;
  wire        t_y_out_2, t_y_out_1;
  reg         t_x_in, t_clock, t_reset;

  Moore_Model_Fig_5_20  M1  (t_y_out_1, t_x_in, t_clock, t_reset);
  Moore_Model_STR_Fig_5_20  M2  (t_y_out_2, A, B, t_x_in, t_clock, t_reset);

  initial #200 $finish;
  initial begin
        t_reset = 0;
        t_clock = 0;
   #5 t_reset = 1;
   repeat (16)
     #5 t_clock = ~t_clock;
  end
  initial begin
          t_x_in = 0;
   #15 t_x_in = 1;
   repeat (8)
     #10 t_x_in = ~t_x_in;
  end

endmodule
```

## Example 2: 4-bit universal shift register (1/3)

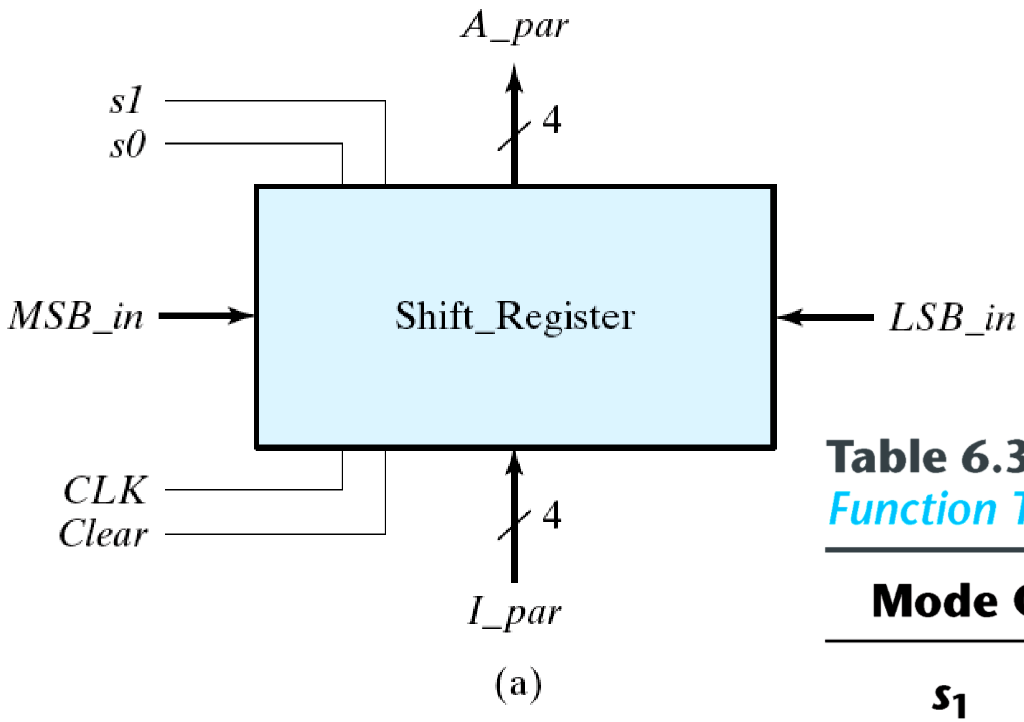


Fig. 6.7
Four-bit universal shift register

**Table 6.3**
*Function Table for the Register of Fig. 6.7*

| Mode Control | | |
|---|---|---|
| $s_1$ | $s_0$ | **Register Operation** |
| 0 | 0 | No change |
| 0 | 1 | Shift right |
| 1 | 0 | Shift left |
| 1 | 1 | Parallel load |

# Example 2: 4-bit universal shift register (2/3)

## Behavioral model

```
// Behavioral description of a 4-bit universal shift register
// Fig. 6.7 and Table 6.3

module Shift_Register_4_beh (
  output reg  [3: 0]   A_par,              // Register output
  input       [3: 0]   I_par,              // Parallel input
  input                s1, s0,             // Select inputs
                       MSB_in, LSB_in,     // Serial inputs
                       CLK, Clear_b        // Clock and Clear_b
);

 always @ (posedge CLK, negedge Clear_b)
     if (~Clear_b) A_par <= 4'b0000;
     else
        case ({s1, s0})
          2'b00: A_par <= A_par;                   // No change
          2'b01: A_par <= {MSB_in, A_par[3: 1]};   // Shift right
          2'b10: A_par <= {A_par[2: 0], LSB_in};   // Shift left
          2'b11: A_par <= I_par;                   // Parallel load of input
        endcase
endmodule
```

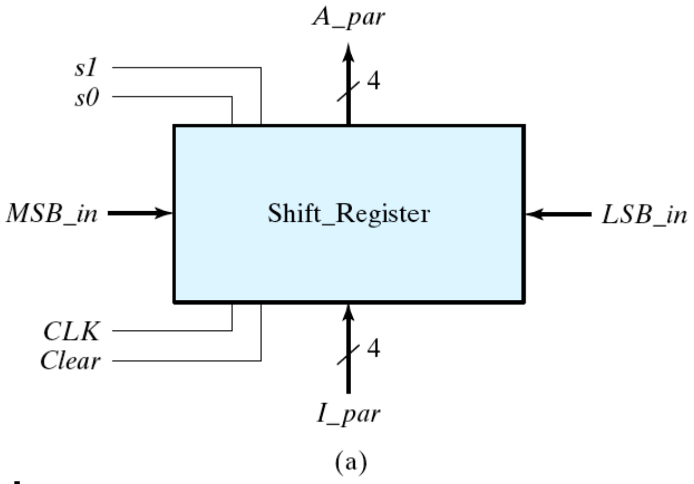


**Table 6.3**
*Function Table for the Register of Fig. 6.7*

| Mode Control | | |
|---|---|---|
| $s_1$ | $s_0$ | Register Operation |
| 0 | 0 | No change |
| 0 | 1 | Shift right |
| 1 | 0 | Shift left |
| 1 | 1 | Parallel load |

## Example 2: 4-bit universal shift register (3/3)

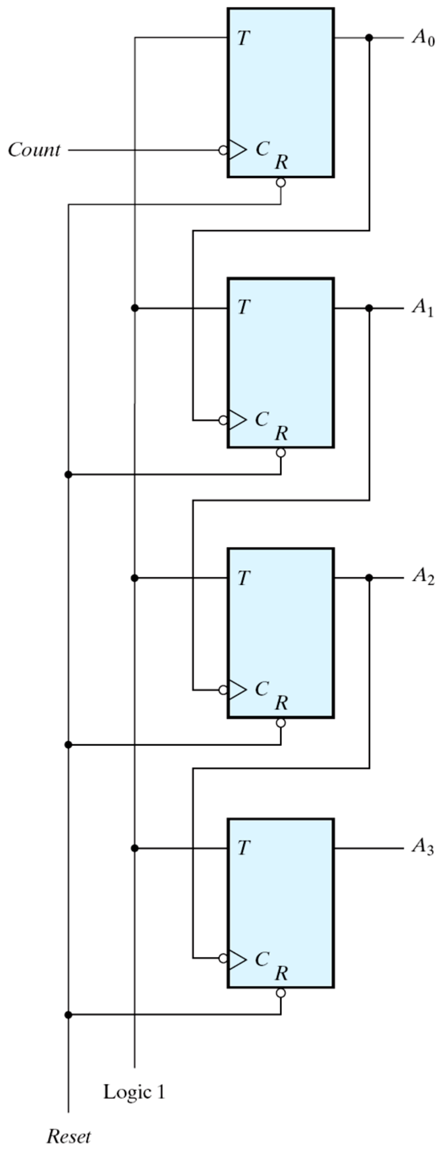
```
module t_Shift_Register_4_beh ();
   reg                    s1, s0,                  // Select inputs
                          MSB_in, LSB_in,          // Serial inputs
                          clk, reset_b;            // Clock and Clear_b
   reg        [3: 0]      I_par;                   // Parallel input
   wire       [3: 0]      A_par;                   // Register output

   Shift_Register_4_beh M0 (A_par, I_par,s1, s0, MSB_in, LSB_in, clk, reset_b);

   initial #200 $finish;
   initial begin clk = 0; forever #5 clk = ~clk; end

   initial fork
      // test reset action load
      #3 reset_b = 1;
      #4 reset_b = 0;
      #9 reset_b = 1;

      // test parallel load
      #10 I_par = 4'hA;
      #10 {s1, s0} = 2'b11;

      // test shift right
      #30 MSB_in = 1'b0;
      #30 {s1, s0} = 2‘b01;

      // test shift left
      #80 LSB_in = 1'b1;
      #80 {s1, s0} = 2'b10;

      // test circulation of data
      #130 {s1, s0} = 2'b11;
      #140 {s1, s0} = 2'b00;

     // test reset on the fly
      #150 reset_b = 1'b0;
      #160 reset_b = 1'b1;
      #160 {s1, s0} = 2'b11;

   join
endmodule
```

## Example 3: 4-bit ripple counter (1/2)

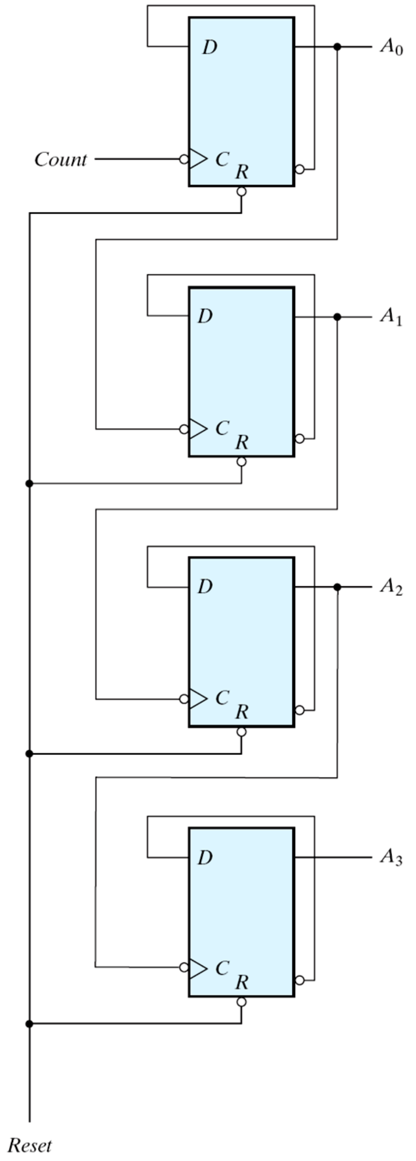
**Table 6.4**
*Binary Count Sequence*

| $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |

Fig. 6.8
Four-bit binary ripple counter



(a) With *T* flip-flops



(b) With *D* flip-flops

## Example 3: 4-bit ripple counter (2/2)

### Structural model

```
`timescale 1ns / 100 ps
module Ripple_Counter_4bit (A3,A2,A1,A0, Count, Reset);
  output A3,A2,A1,A0;
  input Count,Reset;
//Instantiate complementing flip-flop
   Comp_D_flip_flop F0 (A0, Count, Reset);
   Comp_D_flip_flop F1 (A1, A0, Reset);
   Comp_D_flip_flop F2 (A2, A1, Reset);
   Comp_D_flip_flop F3 (A3, A2, Reset);
endmodule

//Complementing flip-flop with delay
//Input to D flip-flop = Q'
module Comp_D_flip_flop (Q, CLK, Reset);
   output Q;
   input CLK, Reset;
   reg Q;
   always @ (negedge CLK, posedge Reset)
   if (Reset) Q <= 1’b0; else Q <= #2 ~Q;
endmodule
```

```
//Stimulus for testing ripple counter
module testcounter;
   reg Count;
   reg Reset;
   wire A0,A1,A2,A3;
//Instantiate ripple counter
   Ripple_Counter_4bit M0 (A3, A2, A1, A0, Count, Reset);
   always
   #5 Count = ~Count;
   initial
     begin
        Count = 1'b0;
        Reset = 1'b1;
        #4 Reset = 1'b0;
     end

   initial #170 $finish;

endmodule
```

## Exercise 1: Four-bit universal shift register

- Design and verify the four-bit universal shift register using Verilog HDL
  - structural (gate-level) modeling

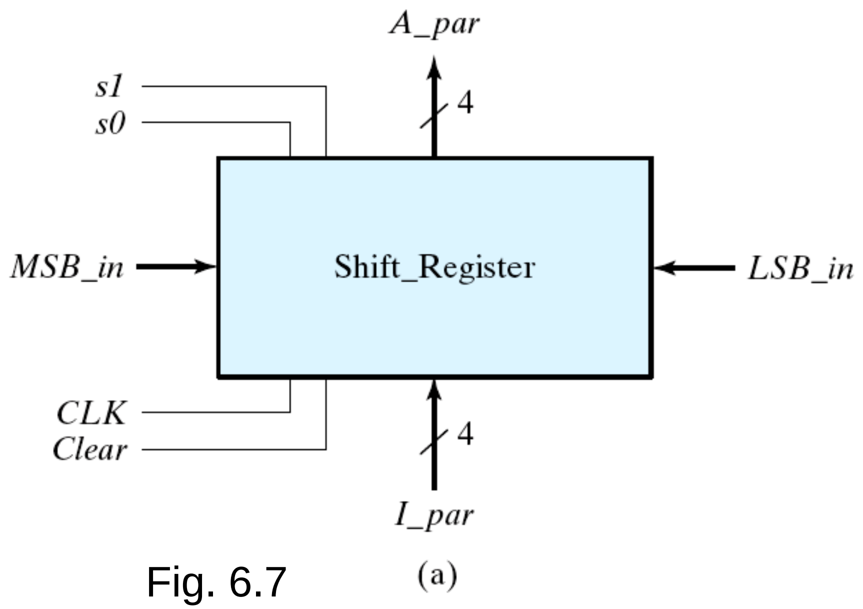


Fig. 6.7
Four-bit universal shift register

**Table 6.3**
*Function Table for the Register of Fig. 6.7*

| Mode Control | | |
|---|---|---|
| $s_1$ | $s_0$ | **Register Operation** |
| 0 | 0 | No change |
| 0 | 1 | Shift right |
| 1 | 0 | Shift left |
| 1 | 1 | Parallel load |

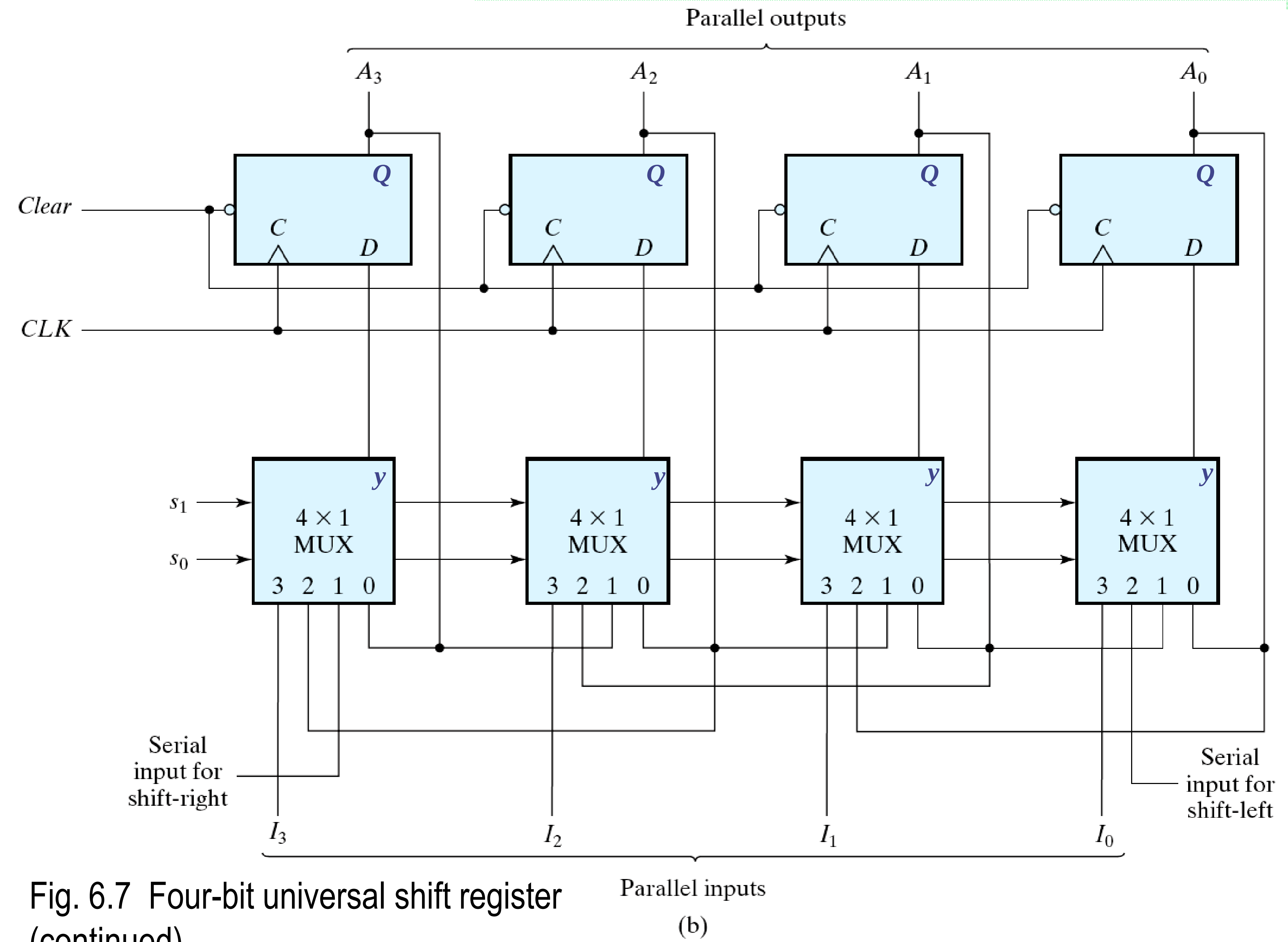


Fig. 6.7 Four-bit universal shift register (continued)

## Exercise 2: BCD synchronous counter (1/2)

- Design and verify the BCD synchronous counter using T flip-flops with asynchronous reset and Verilog HDL
  - structural (gate-level) modeling

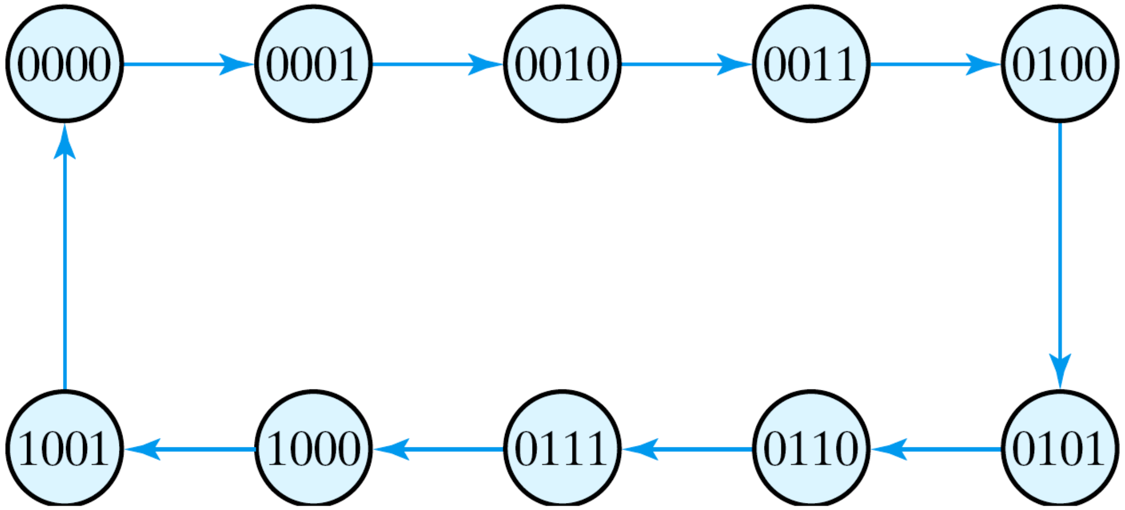


Fig. 6.9 State diagram of a decimal BCD counter

# Exercise 2: BCD synchronous counter (2/2)

- Simplified functions:

$$T_{Q1} = 1$$
$$T_{Q2} = Q_8'Q_1$$

$$T_{Q4} = Q_2Q_1$$
$$T_{Q8} = Q_8Q_1 + Q_4Q_2Q_1$$
$$y = Q_8Q_1$$

**Table 6.5**
*State Table for BCD Counter*

| Present State | | | | Next State | | | | Output | Flip-Flop Inputs | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_8$ | $Q_4$ | $Q_2$ | $Q_1$ | $Q_8$ | $Q_4$ | $Q_2$ | $Q_1$ | $y$ | $TQ_8$ | $TQ_4$ | $TQ_2$ | $TQ_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |