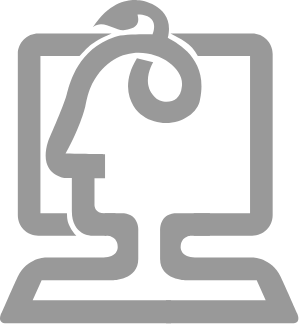


**Digital Logic Design Project**

****

AmirKabir University (AUT-CE)

December 2024

Professor: Dr. Seddighi

Authors: Mohammad Javad Akbari, Parsa Asadi

**In the name of God**

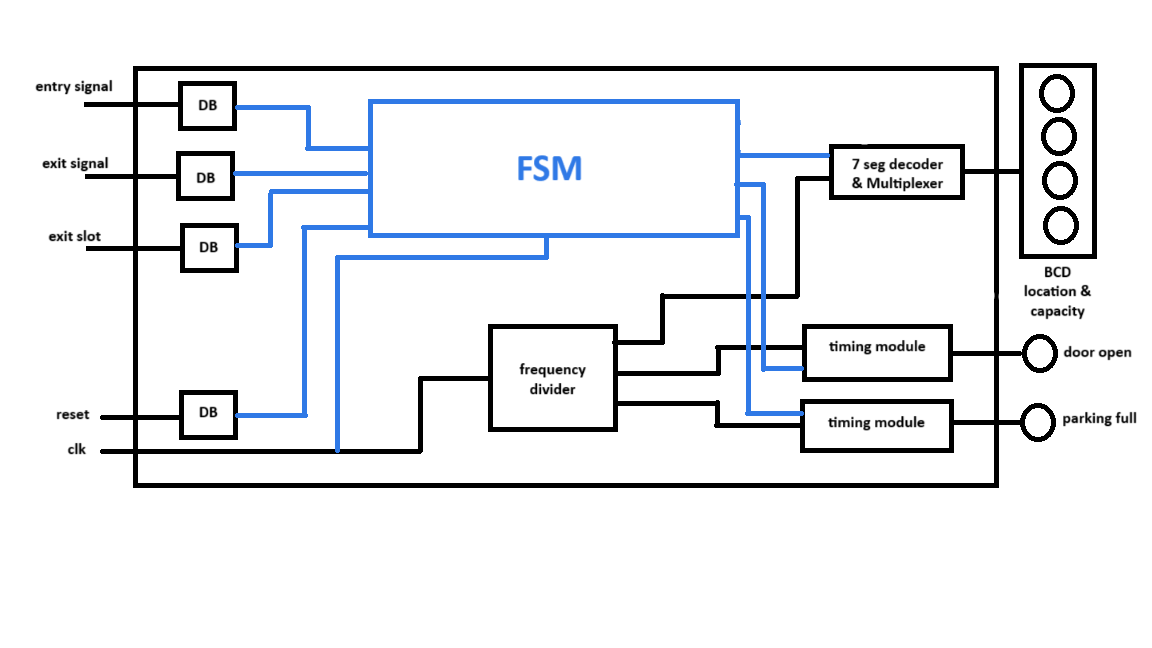
**Introduction**

This project is designed to implement a car parking system using Verilog. In this project, at the entrance of the parking system, a sensor is used to detect the presence of a vehicle. Once the sensor is triggered and there are some available parking slots, the gate opens to let the vehicle in, otherwise, the gate remains locked. This project is implemented on Spartan 3 board.

**Inputs and Outputs**

|  |  |
| --- | --- |
| Inputs | Outputs |
| clk | is\_full |
| entry\_sensor | is\_open |
| exit\_sensor | capacity [2:0] |
| exit\_slot [1:0] | location [1:0] |
| reset (active low) | spots [3:0] |

**System Architecture**

****

**State Diagram (Moore machine)**

This FSM works in 5 main states based on capacity: S0 (Full), S1 (4 cases), S2 (6 cases), S3 (4 cases), S4 (Idle).

entry\_sensor

exit\_sensor & exit\_slot

reset

Note that this is a simplified diagram. In practice, the machine has 16 cases in total based on parking slots. Output in each state is handled based on memory registers. Some arrows are removed due to simplicity.

**FSM in Verilog**

1. module FSM(

2. input clk, // Clock signal

3. input reset, // Reset signal (active low)

4. input entry\_signal, // Signal when a car enters

5. input exit\_signal, // Signal when a exits

6. input [1:0] exit\_slot, // Selects exiting spot

7. output reg is\_open, // Signal to open the door

8. output reg is\_full, // Signal when parking is full

9. output reg [3:0] spots, // Occupied spots

10. output reg [2:0] capacity,// Current remaining capacity

11. output reg [1:0] location // First available empty slot

12. );

13.

14. // Parameters for states based on remaining capacity

15. parameter S4 = 3'b100; // 4 slots remaining (Idle)

16. parameter S3 = 3'b011; // 3 slots remaining

17. parameter S2 = 3'b010; // 2 slots remaining

18. parameter S1 = 3'b001; // 1 slot remaining

19. parameter S0 = 3'b000; // 0 slots remaining (Full)

20.

21. // State change logic

22. always @(posedge clk or reset) begin

23. if (~reset) begin

24.

25. capacity = S4;

26. spots = 4'b0000; // All spots are free (0 = free, 1 = occupied)

27. is\_full = 0;

28. is\_open = 0;

29. location = 2'b00; // Default to the first slot

30.

31. end else begin

32.

33. is\_open = 0;

34. is\_full = 0;

35.

36. // Parking management logic

37. case (capacity)

38. S3, S2, S1: begin

39. if (entry\_signal) begin

40. // Find the first available spot

41. if (spots[0] == 1'b0) location = 2'b00;

42. else if (spots[1] == 1'b0) location = 2'b01;

43. else if (spots[2] == 1'b0) location = 2'b10;

44. else if (spots[3] == 1'b0) location = 2'b11;

45.

46. // fill the spot

47. spots[location] = 1'b1;

48. is\_open = 1;

49.

50. end else if (exit\_signal) begin

51.

52. // Free the selected spot

53. spots[exit\_slot] = 1'b0;

54. is\_open = 1;

55.

56. end

57. end

58.

59. S4: begin

60. if (entry\_signal) begin

61. // fill the first spot

62. spots[0] = 1'b1;

63. is\_open = 1;

64. end

65. end

66.

67. S0: begin

68. if (exit\_signal) begin

69. // Free the selected spot

70. spots[exit\_slot] = 1'b0;

71. is\_open = 1;

72. end

73. end

74.

75. endcase

76.

77. // Next state logic

78. case (capacity)

79. S4: begin

80. if (entry\_signal)

81. capacity = S3;

82. end

83.

84. S3: begin

85. if (entry\_signal)

86. capacity = S2;

87. else if (exit\_signal)

88. capacity = S4;

89. end

90.

91. S2: begin

92. if (entry\_signal)

93. capacity = S1;

94. else if (exit\_signal)

95. capacity = S3;

96. end

97.

98. S1: begin

99. if (entry\_signal)

100. capacity = S0;

101. else if (exit\_signal)

102. capacity = S2;

103. end

104.

105. S0: begin

106. is\_full = 1; // Parking is full

107. if (exit\_signal)

108. capacity = S1;

109. end

110.

111. endcase

112.

113. end

114. end

115.

116. endmodule

117.

**FSM testbench in Verilog**

1. module FSM\_TB;

2.

3. // Inputs

4. reg clk;

5. reg reset;

6. reg entry\_signal;

7. reg exit\_signal;

8. reg [1:0] exit\_slot;

9.

10. // Outputs

11. wire is\_open;

12. wire is\_full;

13. wire [3:0] spots;

14. wire [2:0] capacity;

15. wire [1:0] location;

16.

17. // Instantiate the ParkingSystem module

18. FSM uut (

19. .clk(clk),

20. .reset(reset),

21. .entry\_signal(entry\_signal),

22. .exit\_signal(exit\_signal),

23. .exit\_slot(exit\_slot),

24. .is\_open(is\_open),

25. .is\_full(is\_full),

26. .spots(spots),

27. .capacity(capacity),

28. .location(location)

29. );

30.

31. // Clock generation

32. always #5 clk = ~clk;

33.

34. // Test sequence

35. initial begin

36. // Initialize inputs

37. clk = 0;

38. reset = 0;

39. entry\_signal = 0;

40. exit\_signal = 0;

41. exit\_slot = 2'b00;

42.

43. // Enable machine

44. #10 reset = 1;

45.

46. // Test case 1: First car enters

47. #10 entry\_signal = 1;

48. #10 entry\_signal = 0; // Simulate a short entry pulse

49.

50. // Test case 2: Second car enters

51. #10 entry\_signal = 1;

52. #10 entry\_signal = 0;

53.

54. // Test case 3: Third car enters

55. #10 entry\_signal = 1;

56. #10 entry\_signal = 0;

57.

58. // Test case 4: Fourth car enters (parking is full)

59. #10 entry\_signal = 1;

60. #10 entry\_signal = 0;

61.

62. // Check is\_full signal

63. #10;

64.

65. // Test case 5: A car exits from slot 2

66. #10 exit\_signal = 1;

67. exit\_slot = 2'b10; // Specify the slot

68. #10 exit\_signal = 0;

69.

70. // Test case 6: Another car enters

71. #10 entry\_signal = 1;

72. #10 entry\_signal = 0;

73.

74. #50;

75. $finish;

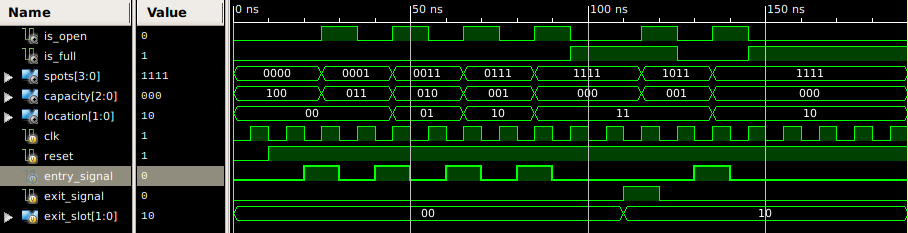
76. end

77.

78. endmodule

79.

**FSM testbench simulation result**

****