EENG 284 – Digital Design

Lab 3- Rock, Paper, Scissors

Solutions

# onesToDense Module:

Table 1: The truth table for the onesToDense function.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| r | p | s | play | Note |
| 0 | 0 | 0 | 11 | Invalid |
| 0 | 0 | 1 | 10 | Scissors |
| 0 | 1 | 0 | 01 | Paper |
| 0 | 1 | 1 | 11 | Invalid |
| 1 | 0 | 0 | 00 | Rock |
| 1 | 0 | 1 | 11 | Invalid |
| 1 | 1 | 0 | 11 | Invalid |
| 1 | 1 | 1 | 11 | Invalid |

play[1] = r’p’s’+r’p’s+r’ps+rp’s+rps’+rps

play[0] = r’p’s’+r’ps’+r’ps+rp’s+rps’+rps

Verilog

module onesToDense (throw, play);

output wire [1:0] play;

input wire [2:0] throw;

wire r, p, s;

assign r = throw[2];

assign p = throw[1];

assign s = throw[0];

assign play[1] =

assign play[0] =

endmodule

# playToSeven Module:

Table 2: Table to determine the bit values for the 7-segment display LEDs to produce the throw patterns.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| pPlay | 6 | 5 | 4 | 3 | 2 | 1 | 0 | sevenSeg | Note |
| 00 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 7’b0100011 | Rock |
| 01 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 7’b1000000 | Paper |
| 10 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 7’b0001001 | Scissors |
| 11 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7’b1111111 | Invalid |

Verilog

module playToSeven (pPlay, sevenSeg);

output reg [6:0] sevenSeg;

input wire [1:0] pPlay;

always @(\*)

case (pPlay)

2'b00: sevenSeg = 7'b0100011; // rock

2'b01: sevenSeg = 7'b1000000; // paper

2'b10: sevenSeg = 7'b0001001; // scissors

2'b11: sevenSeg = 7'b1111111; // invalid

endcase

endmodule

# winLose Module:

Table 4: Abbreviated truth table for the winLose module.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| button | p1Play | p2Play | sevenSeg | Note |
| 0 | 00 (rock) | 00 (rock) | 7’b0100001 | Draw |
| 0 | 00 (rock) | 01 (paper) | 7’b0100100 | 2 |
| 0 | 00(rock) | 10 (scissor) | 7’b1001111 | 1 |
| 0 | 00(rock) | 11 (invalid) | 7’b1001111 | 1 |
| 0 | 01 (paper) | 00 (rock) | 7’b0100100 | 1 |
| 0 | 01 (paper) | 01 (paper) | 7’b0100001 | Draw |
| 0 | 01 (paper) | 10 (scissor) | 7’b0100100 | 2 |
| 0 | 01 (paper) | 11 (invalid) | 7’b1001111 | 1 |
| 0 | 10 (scissors) | 00 (rock) | 7’b0100100 | 2 |
| 0 | 10 (scissors) | 01 (paper) | 7’b1001111 | 1 |
| 0 | 10 (scissors) | 10 (scissor) | 7’b0100001 | Draw |
| 0 | 10 (scissors) | 11 (invalid) | 7’b1001111 | 1 |
| 0 | 11 (invalid) | 00 (rock) | 7’b0100100 | 2 |
| 0 | 11 (invalid) | 01 (paper) | 7’b0100100 | 2 |
| 0 | 11 (invalid) | 10 (scissor) | 7’b0100100 | 2 |
| 0 | 11 (invalid) | 11 (invalid) | 7’b0100001 | Draw |
| 1 | xx (don’t care) | xx (don’t care) | 7’b1111111 | Blank |

Verilog

module winLose(p1Play, p2Play, playButton, sevenSeg);

output reg [6:0] sevenSeg;

input wire [1:0] p1Play, p2Play;

input wire playButton;

always @(\*)

case ({playButton, p1Play, p2Play})

5'b00000: sevenSeg = 7'b0100001; // R R D

5'b00001: sevenSeg = 7'b0100100; // R P 2

5'b00010: sevenSeg = 7'b1001111; // R S 1

5'b00011: sevenSeg = 7'b1001111; // R V 1

5'b00100: sevenSeg = 7'b1001111; // P R 1

5'b00101: sevenSeg = 7'b0100001; // P P D

5'b00110: sevenSeg = 7'b0100100; // P S 2

5'b00111: sevenSeg = 7'b1001111; // P V 1

5'b01000: sevenSeg = 7'b0100100; // S R 2

5'b01001: sevenSeg = 7'b1001111; // S P 1

5'b01010: sevenSeg = 7'b0100001; // S S D

5'b01011: sevenSeg = 7'b1001111; // S V 1

5'b01100: sevenSeg = 7'b0100100; // V R 2

5'b01101: sevenSeg = 7'b0100100; // V P 2

5'b01110: sevenSeg = 7'b0100100; // V S 2

5'b01111: sevenSeg = 7'b0100001; // V V D

default: sevenSeg = 7'b1111111; // Blank

endcase

endmodule

# rpsGame Module:

Verilog

module rpsGame(p1Throw, p1SevenSeg, p2Throw, p2SevenSeg, playButton, winLoseSeg);

input wire [2:0] p1Throw, p2Throw;

input wire playButton;

output wire [6:0] p1SevenSeg, p2SevenSeg, winLoseSeg;

wire [1:0] p1Dense, p2Dense;

onesToDense p1o2d(p1Throw, p1Dense);

onesToDense p2o2d(p2Throw, p2Dense);

playToSeven p1p2s(p1Dense, p1SevenSeg);

playToSeven p2p2s(p2Dense, p2SevenSeg);

winLose (p1Dense, p2Dense, playButton, winLoseSeg);

endmodule

# Pin Assignment:

|  |  |  |  |
| --- | --- | --- | --- |
| Segment | Player 1 Throw | Player 2 Throw | Win/Lose |
| seg[6] | PIN\_Y18 | PIN\_AC22 | PIN\_AF24 |
| seg[5] | PIN\_Y19 | PIN\_AC23 | PIN\_AC19 |
| seg[4] | PIN\_Y20 | PIN\_AC24 | PIN\_AE25 |
| seg[3] | PIN\_W18 | PIN\_AA22 | PIN\_AE26 |
| seg[2] | PIN\_V17 | PIN\_AA23 | PIN\_AB19 |
| seg[1] | PIN\_V18 | PIN\_Y23 | PIN\_AD26 |
| seg[0] | PIN\_V19 | PIN\_Y24 | PIN\_AA18 |

|  |  |  |
| --- | --- | --- |
|  | Player 1 Slide Switch | Player 2 Slide Switch |
| slide[2] | PIN\_AD13 | PIN\_AE19 |
| slide[1] | PIN\_AE10 | PIN\_Y11 |
| slide[0] | PIN\_AC9 | PIN\_AC10 |

Section 4.2

Each push-button switch provides a high logic level when it is not pressed, and provides a low logic level when pressed.

|  |  |  |
| --- | --- | --- |
| Play Button | Key[0] | PIN\_P11 |