EENG 284

–

Digital Design Lab

Lab10

–

Stopwatch Control Unit

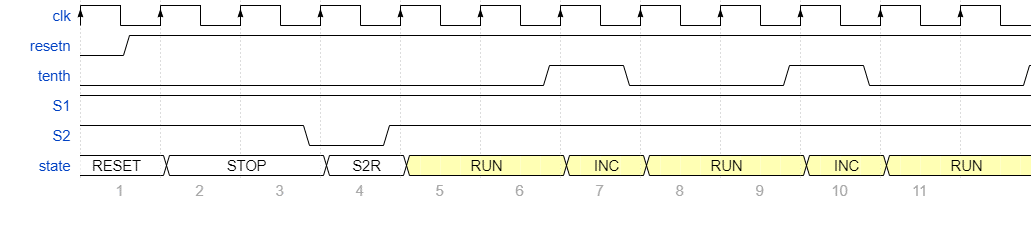
Lab Solutions

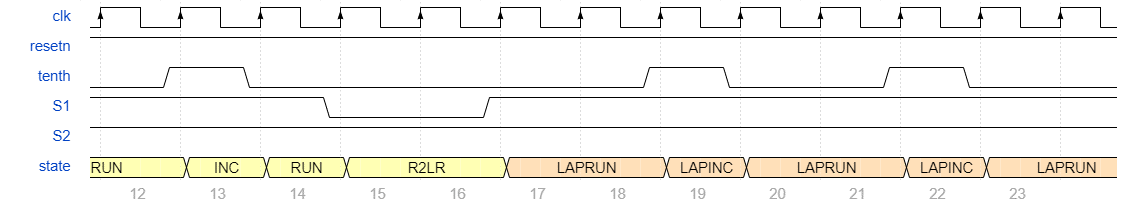
Completed Table 1

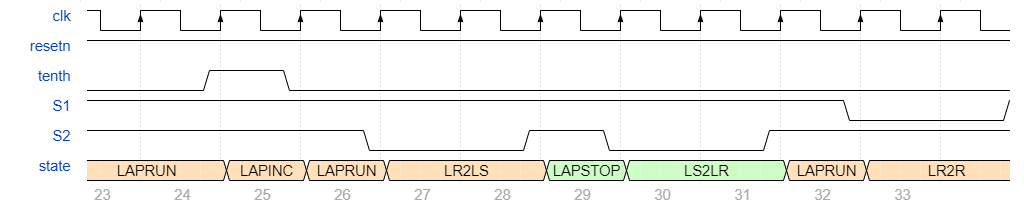
Table 1: Control word table for the stopwatch finite state machine shown in Figure 3.

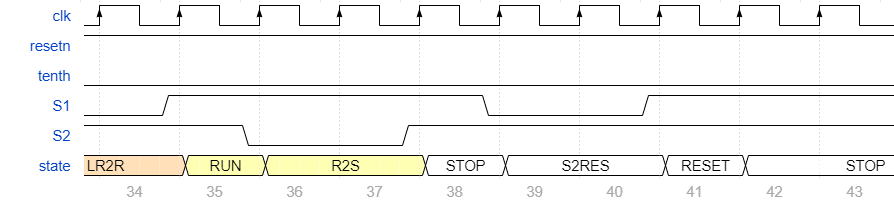
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | cw[5]  2x1 mux | cw[4]  lap register | cw[3]  mod 10 reset | cw[2]  mod10 count | cw[1:0]  timer counter |
|  | 0 = mod10 | 1 = load | 1 = reset | 1 = count up | 11 = load |
|  | 1 = register | 0 = hold | 0 = hold | 0 = hold | 10 =count up |
|  |  |  |  |  | 01 = not used |
|  |  |  |  |  | 00 = hold |
| RESET | 0 | 0 | 1 | 0 | 11 |
| STOP | 0 | 0 | 0 | 0 | 00 |
| S2RES | 0 | 0 | 0 | 0 | 00 |
| S2R | 0 | 0 | 0 | 0 | 00 |
| RUN | 0 | 0 | 0 | 0 | 10 |
| R2LR | 0 | 1 | 0 | 0 | 00 |
| R2S | 0 | 0 | 0 | 0 | 00 |
| INC | 0 | 0 | 0 | 1 | 11 |
| LAPRUN | 1 | 0 | 0 | 0 | 10 |
| LR2R | 1 | 0 | 0 | 0 | 00 |
| LR2LS | 1 | 0 | 0 | 0 | 00 |
| LAPINC | 1 | 0 | 0 | 1 | 11 |
| LAPSTOP | 1 | 0 | 0 | 0 | 00 |
| LS2R | 1 | 0 | 0 | 0 | 00 |
| LS2LR | 1 | 0 | 0 | 0 | 00 |

Completed Figure 6 through Figure 9.









module controlUnit(clk, resetn, cw, sw);

parameter SW\_VECTOR\_LENGTH = 3;

parameter CW\_VECTOR\_LENGTH = 6;

input wire clk, resetn;

output reg [CW\_VECTOR\_LENGTH - 1 : 0] cw;

input wire [SW\_VECTOR\_LENGTH - 1 : 0] sw;

wire tenth, S1, S2;

reg [3:0] state;

reg [3:0] nextstate;

localparam RESET\_CW = 6'b001011;

localparam STOP\_CW = 6'b000000;

localparam S2RESET\_CW = 6'b000000;

localparam S2R\_CW = 6'b000000;

localparam RUN\_CW = 6'b000010;

localparam R2LR\_CW = 6'b010000;

localparam R2S\_CW = 6'b000000;

localparam INC\_CW = 6'b000111;

localparam LAPRUN\_CW = 6'b100010;

localparam LR2R\_CW = 6'b100000;

localparam LR2LS\_CW = 6'b100000;

localparam LAPINC\_CW = 6'b100111;

localparam LAPSTOP\_CW = 6'b100000;

localparam LS2R\_CW = 6'b100000;

localparam LS2LR\_CW = 6'b100000;

parameter RESET\_STATE = 4'b0000,

STOP\_STATE = 4'b0010,

S2RESET\_STATE = 4'b0001,

S2R\_STATE = 4'b0011,

RUN\_STATE = 4'b0100,

R2LR\_STATE = 4'b0110,

R2S\_STATE = 4'b0101,

INC\_STATE = 4'b0111,

LAPRUN\_STATE = 4'b1000,

LR2R\_STATE = 4'b1001,

LR2LS\_STATE = 4'b1010,

LAPINC\_STATE = 4'b1011,

LAPSTOP\_STATE = 4'b1100,

LS2R\_STATE = 4'b1101,

LS2LR\_STATE = 4'b1110;

assign tenth = sw[0];

assign S1 = sw[1];

assign S2 = sw[2];

// Take care of reset logic

always @(negedge resetn, posedge clk)

if (!resetn)

state <= RESET\_STATE;

else

state <= nextstate;

// Determine the output

always @(state) // always block to compute output

begin

case(state)

RESET\_STATE: cw = RESET\_CW;

S2RESET\_STATE: cw = S2RESET\_CW;

STOP\_STATE: cw = STOP\_CW;

s2R\_STATE: cw = S2R\_CW;

RUN\_STATE: cw = RUN\_CW;

R2LR\_STATE: cw = R2LR\_CW;

R2S\_STATE: cw = R2S\_CW;

INC\_STATE: cw = INC\_CW;

LAPINC\_STATE: cw = LAPINC\_CW;

LAPRUN\_STATE: cw = LAPRUN\_CW;

LR2R\_STATE: cw = LR2R\_CW;

LR2LS\_STATE: cw = LR2LS\_CW

LAPSTOP\_STATE: cw = LAPSTOP\_CW;

LS2R\_STATE: cw = LS2R\_CW;

LS2LR\_STATE: cw = LS2LR\_CW;

default: cw = RESET\_CW;

endcase

end

//------------------------------------------

// Determine the next state

// buttons are logic 0 when pressed

//------------------------------------------

always @(\*) // always block to compute nextstate

begin

case(state)

RESET\_STATE:

nextstate = STOP\_STATE;

STOP\_STATE:

begin

case({S2,S1})

2'b10: nextstate = S2RESET\_STATE;

2'b01: nextstate = S2R\_STATE;

default: nextstate = STOP\_STATE;

endcase

end

S2RESET\_STATE:

begin

case (S1)

1'b1: nextstate = RESET\_STATE;

default: nextstate = S2RESET\_STATE;

endcase

end

S2R\_STATE:

begin

case(S2)

1'b1: nextstate = RUN\_STATE;

default: nextstate = S2R\_STATE;

endcase

end

RUN\_STATE:

begin

case({tenth, S2, S1})

3'b111: nextstate = INC\_STATE;

3'b010: nextstate = R2LR\_STATE;

3'b001: nextstate = R2S\_STATE;

default: nextstate = RUN\_STATE;

endcase

end

R2LR\_STATE:

begin

case(S1)

1'b1: nextstate = LAPRUN\_STATE;

default: nextstate = R2LR\_STATE;

endcase

end

R2S\_STATE:

begin

case(S2)

1'b1: nextstate = STOP\_STATE;

default: nextstate = R2S\_STATE;

endcase

end

INC\_STATE:

begin

nextstate = RUN\_STATE;

end

LAPRUN\_STATE:

begin

case({tenth, S2, S1})

3'b111: nextstate = LAPINC\_STATE;

3'b010: nextstate = LR2R\_STATE;

3'b001: nextstate = LR2LS\_STATE;

default: nextstate = LAPRUN\_STATE;

endcase

end

LR2R\_STATE:

begin

case(S1)

1'b1: nextstate = RUN\_STATE;

default: nextstate = LR2R\_STATE;

endcase

end

LR2LS\_STATE:

begin

case(S2)

1'b1: nextstate = LAPSTOP\_STATE;

default: nextstate = LR2LS\_STATE;

endcase

end

LAPINC\_STATE:

begin

nextstate = LAPRUN\_STATE;

end

LAPSTOP\_STATE:

begin

case ({S2, S1})

2'b10: nextstate = LS2R\_STATE;

2'b01: nextstate = LS2LR\_STATE;

default: nextstate = LAPSTOP\_STATE;

endcase

end

LS2R\_STATE:

begin

case(S1)

1'b1: nextstate = RESET\_STATE;

default: nextstate = LS2R\_STATE;

endcase

end

LS2LR\_STATE:

begin

case (S2)

1'b1: nextstate = LAPRUN\_STATE;

default: nextstate = LS2LR\_STATE;

endcase

end

default: nextstate = RESET\_STATE;

endcase

end

endmodule

* Produce a timing diagram with the following characteristics. Zoom to fill the available horizontal space with the waveform.

