FH Vorarlberg
University of Applied Sciences

Assignment 03

Debouncer

Embedded Systems 3

Fachhochschule Vorarlberg

Master Mechatronics

Eingereicht bei

Dr. Andrè Mitterbacher

Vorgelegt von

Roman Passler

Dornbirn, 07.01.2018

Inhaltsverzeichnis

Abbildungsverzeichnis **II**

Tabellenverzeichnis **III**

Listings **IV**

1 Debouncer **1**
1.1 Einleitung 1
1.2 Implementierung 1
1.3 Test Bench 3
1.4 Simulationsscript 5
1.5 Transkript und Waveform Window 6
1.6 Vor- und Nachteile der Implementierung 6

Abbildungsverzeichnis

1.1 Waveform Window 6

Tabellenverzeichnis

Listings

1.1 Implementierung 1
1.2 Testbench für den Up / Down Counter 3
1.3 Simulationsscript 5

1 Debouncer

1.1 Einleitung

Der Switch, der für den Betrieb der Stoppuhr bestimmt ist, muss entprellt werden, da ansonsten die Statemachine nicht ordnungsgemäß funktioniert. Dieses Beispiel entprellt die „high“ sowie „low“ Flanke.

- Wenn der Switch „low“ ist, wird der erste Counter gestartet von einem Initialwert (alle Bits auf eins). Das Signal ist solange gültig, bis der Counter null erreicht hat. Wenn der Switch „high“ schaltet, so wird der erste Counter sofort rückgesetzt auf seinen Initialwert.
- Wenn der Switch „high“ ist, wird der zweite Counter gestartet von einem Initialwert (alle Bits auf eins). Das Signal ist solange gültig, bis der Counter null erreicht hat. Wenn der Switch „low“ schaltet, so wird der zweite Counter sofort rückgesetzt auf seinen Initialwert.
- Die Bitbreite des Counters muss in einem Header File mit einem Makro („BITS“) definiert werden. Für diese Implementierung wird „BITS“ auf drei gesetzt.
- Die Flankenerkennung der fallenden und steigenden Flanke des entprellten Signals wird als „sw_hi“ und „sw_lo“ ausgegeben. Beide Ausgänge sind nur für einen Clock Zyklus auf „high“.

1.2 Implementierung

Die Implementierung erfolgt wie in Listing 1.1 dargestellt ist. In Codezeile **??** wird der Ausgang Counter „cnt“ mit variabler Bitlänge definiert definiert.

```
1  /*-----------------------------------------
2  Project:    debounce
3  Purpose:
4  Author:     rpa2306
5  Version:    00, 01.06.2018
6  -----------------------------------------*/
7
8  `include "debounce.sh"
9
10 module debounce (
11     input   logic           rst_n,
12     input   logic           clk50m,
13     input   logic           sw,
14     output  logic           sw_hi,
15     output  logic           sw_lo,
16     output  logic           sw_dbnc
```

```
17  );
18
19  // (1) counter for high
20      logic [`BITS-1:0]    sw_hi_cnt;
21      logic                sw_hi_cnt_zero;
22
23  // (2) counter for low
24      logic [`BITS-1:0]    sw_lo_cnt;
25      logic                sw_lo_cnt_zero;
26
27  always_ff @(negedge rst_n or posedge clk50m) begin
28      // reset
29      if(!rst_n) begin
30          sw_hi            <= 1'b0;
31          sw_lo            <= 1'b0;
32          sw_dbnc          <= 1'b0;
33          sw_lo_cnt_zero   <= 1'b0;
34          sw_hi_cnt_zero   <= 1'b0;
35          sw_hi_cnt        <= '0;
36          sw_lo_cnt        <= '1;
37      end
38      else if (sw && !sw_hi_cnt) begin
39          sw_lo_cnt        <= '1;
40          sw_hi            <= 1'b0;
41          sw_dbnc          <= 1'b1;
42
43      end
44      else if (!sw && !sw_lo_cnt) begin
45          sw_hi_cnt        <= '1;
46          sw_lo            <= 1'b0;
47          sw_dbnc          <= 1'b0;
48      end
49      // counter if someone pressed the switch
50      else if (sw) begin
51          sw_hi_cnt        <= sw_hi_cnt - `BITS'd1;
52          // init low counter to highest value
53          sw_lo_cnt        <= '1;
54          sw_lo_cnt_zero   <= 1'b0;
55
56          if (sw_hi_cnt == `BITS'd1) begin
57              sw_hi            <= 1'b1;
58              sw_hi_cnt_zero   <= 1'b1;
59          end
60      end
61      // counter if someone released the switch
62      else if (!sw) begin
63          sw_lo_cnt        <= sw_lo_cnt - `BITS'd1;
64          // init low counter to highest value
65          sw_hi_cnt        <= '1;
66          sw_hi_cnt_zero   <= 1'b0;
67
68          if (sw_lo_cnt == `BITS'd1) begin
69              sw_lo            <= 1'b1;
70              sw_lo_cnt_zero   <= 1'b1;
71          end
72      end
73
74      // initial condition (startup)
75      else begin
76          sw_hi_cnt    <= '0;
77          sw_lo_cnt    <= '1;
```

```
78      end
79  end
80
81  endmodule
```

Listing 1.1: Implementierung

1.3 Test Bench

In Listing 1.2 ist die Testbench ersichtlich. Es wurde ein automatisiertes Testscript erstellt, welches einen Fehler ausgibt, sobald der Up oder Down Test fehlschlägt (Codezeile ?? bis ?? und ?? bis ??).

```
1   /*-----------------------------------------
2   Project:      debounce
3   Purpose:
4   Author:       rpa2306
5   Version:      00, 01.06.2018
6   -----------------------------------------*/
7
8   module tb_debounce();
9
10  /*
11  *.  (1) Create wires to connect the DUT
12  *.  Like footprints for an IC on a PCB
13  */
14
15
16  logic         rst_n;
17  logic         clk50m;
18  logic         sw;
19
20  logic         sw_hi;
21  logic         sw_lo;
22  logic         sw_dbnc;
23
24  /*
25  *    (2) Create an instance of the DUT
26  */
27
28  debounce        DUT (.*);
29
30  /*
31  *    (3) Create stimuli for all inputs
32  */
33
34  logic run_sim = 1'b1;
35
36  initial begin: clk_gen
37      clk50m = 1'b0;
38
39      while(run_sim) begin
40          #10ns;
41          clk50m = !clk50m;
42      end
43  end
44
```

```
45  initial begin
46
47      $display("-------------------------------");
48      $display("     debounce started     ");
49      $display("-------------------------------");
50      sw = 1'b0;
51      rst_n = 1'b1;
52      #50ns;
53      rst_n = 1'b0;
54      #50ns;
55      rst_n = 1'b1;
56      #50ns;
57
58      @(negedge clk50m);
59      sw = 1'b0;
60      repeat (10) begin
61              @ (posedge clk50m);
62      end
63      $display("\t%b\t%b\t%b\t%b\t%b\t%b",rst_n, clk50m, sw, sw_hi, sw_lo
            , sw_dbnc);
64      #50ns;
65      @(negedge clk50m);
66      sw = 1'b1;
67
68      repeat (5) begin
69              @ (posedge clk50m);
70      end
71      @ (negedge clk50m);
72      sw = 1'b0;
73      //@ (posedge clk50m);
74
75      @ (negedge clk50m);
76      sw = 1'b1;
77      repeat (3) begin
78              @ (posedge clk50m);
79      end
80
81      @(negedge clk50m);
82      sw = 1'b0;
83      repeat (5) begin
84              @ (posedge clk50m);
85      end
86
87      @ (negedge clk50m);
88      sw = 1'b1;
89      repeat (2) begin
90              @ (posedge clk50m);
91      end
92
93      @ (negedge clk50m);
94      sw = 1'b0;
95
96      repeat (8) begin
97              @ (posedge clk50m);
98      end
99
100     #200ns;
101
102     @ (negedge clk50m);
103     sw = 1'b1;
104
```

```
105     repeat (8) begin
106             @ (posedge clk50m);
107     end
108
109     #200ns;
110
111
112     @ (negedge clk50m);
113     sw = 1'b0;
114
115     repeat (8) begin
116             @ (posedge clk50m);
117     end
118
119     #50ns;
120     @ (negedge clk50m);
121     run_sim = 0;
122     rst_n = 1'b0;
123     $display("------------------------------");
124     $display("    debounce finished     ");
125     $display("------------------------------");
126
127 end
128 endmodule
```

Listing 1.2: Testbench für den Up / Down Counter

1.4 Simulationsscript

In Listing 1.3 ist das Simulationsscript dargestellt. Es beinhaltet dieselben Befehle wie in der letzten Lehrveranstaltung, natürlich angepasst an den Debouncer.

```
1  # Create simulation environment
2  vlib work
3  vmap work work
4
5  # Compile desing files -> use file names
6  vlog ../src/debounce.sv
7  # Compile the test bench
8  vlog tb_debounce.sv
9  # Init simulation -> use module name
10 vsim tb_debounce
11 # -r recursive
12 log -r *
13 do wave_tb_debounce.tcl
14
15 # Run simulation
16 run -all
17 # run 100us
18 # Show results
19 view wave
```

Listing 1.3: Simulationsscript

1.5 Transkript und Waveform Window

In Abbildung 1.1 ist das Waveform Window dargestellt. Es zeigt die geforderten Testfälle. Wie zu sehen ist, wurden die Anforderungen erfüllt.

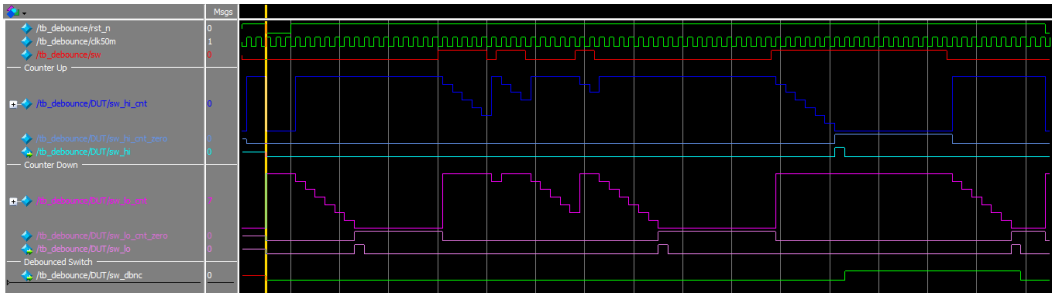

Abbildung 1.1: Waveform Window
Quelle: eigene Ausarbeitung

1.6 Vor- und Nachteile der Implementierung

Folgend sind die Vor- und Nachteile der Implementierung gelistet:

Vorteile

- Variable Bitlänge

Nachteile

- Die Clock Frequenz darf nur so hoch gewählt werden, wie es das „Back Propagation Delay“ erlaubt. Somit ist die maximale Zählfrequenz begrenzt.
- Bei Anwahl des Down-Bits sollte beim Reseten der Counter Wert nicht auf „0“ initalisiert werden, sondern mit dem maximalen Wert

$$2 \ << \ (`counterSize - 1) - 1 \tag{1.1}$$

- Ebenfalls wäre es gut, den Startwert des Counters selbst wählen zu können.