

HÖHERE TECHNISCHE BUNDESLEHRANSTALT HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang: 5BHEL	Gruppe: 2b	Übungsleiter: Prof. DUM
Übungsnummer: II / 1	Übungstitel: IP-Core	
Datum der Übung: 17.11.2015	Teilnehmer: Holzinger, Graussam, Mischling, Schwarzmann	
Datum der Abgabe: 01.12.2015	Schriftführer: Mischling David	Unterschrift:

		Beurteilung
Aufgabenstellung		
Dokumentation		
Messschaltungen		
Messtabellen		
Berechnungen		
Programmlistings		
Auswertung		
Diagramme		
Berechnungen		
Simulationen		
Schlussfolgerungen		
Kommentare		
Inventarliste		
Messprotokoll		
Form		
Summe		

Allgemeiner Teil

Titel der Übung: IP-Core

Übungsnummer: 2/1

Übungsplatz: T7

Datum der Übung: 17.11.2015

Klasse: 5BHEL

Schriftführer: Mischling David

Übungsteilnehmer: Graussam Lukas

Thomas Holzinger

Schwarzmann Philipp

Inhaltsverzeichnis

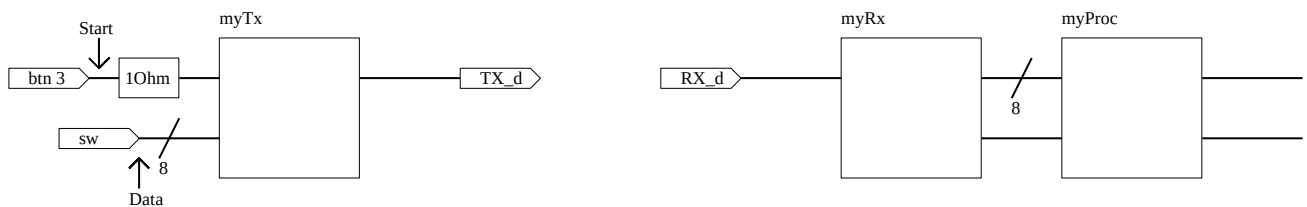
Aufgabenstellung.....	4
Übungsdurchführung	5
Vorbereitung	5
Übung 1	7
Schematic	7
Funktion.....	7
Funktionstest.....	9
Übung 2	10
Schematic	10
Funktion.....	11
Funktionstest.....	12
Erkenntnis.....	13
Inventarliste.....	14
Messprotokoll.....	15

Aufgabenstellung

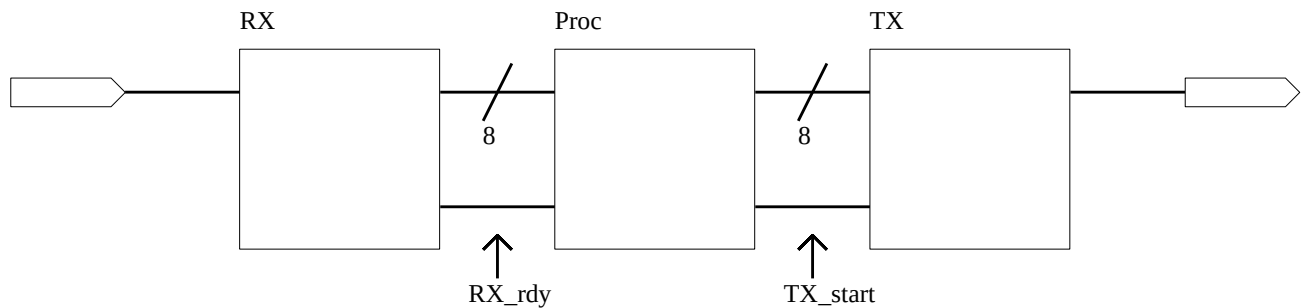
Aufgabe war es, mit den, von dem Übungsleiter zur Verfügung gestellten, Entities ein VHDL-Projekt zu erstellen.

Dieses Projekt sollte eine Datenübertragung simulieren, die anschließend auch am Basys Board getestet werden sollte.

Die zu sendenden Daten sollten über ein Interface (Switches) vom User eingegeben werden, nach einem von den Buttons stammenden Signal gesendet werden, und nach einer Verarbeitung über die LED-7-Segment-Anzeige ausgegeben werden.



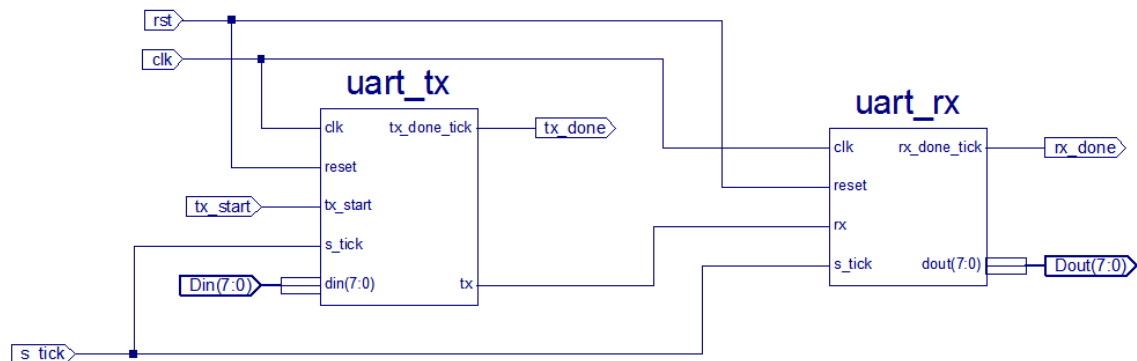
Als Erweiterung sollte ein zweites Projekt erstellt werden, das von dem ersten Programm Daten empfängt, verarbeitet, ausgibt und auch wieder weitersendet.



Übungsdurchführung

Vorbereitung

Bereits vor der Übung wurde zuhause von jedem Schüler ein Xilinx-Projekt angelegt. In diesem wurden die zur Verfügung gestellten Entities verwendet um eine Schemetic zu zeichnen, die eine einfache Datenübertragung simulierte.

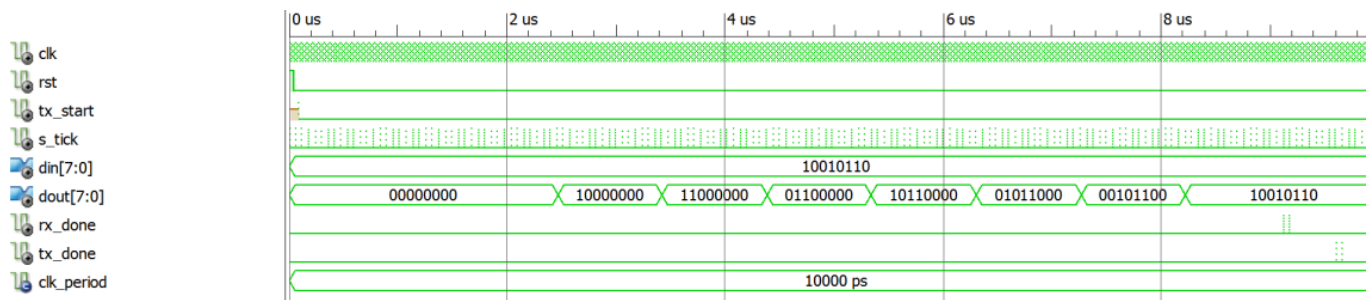


Ebenfalls wurde eine entsprechende Testbench erstellt und somit die Funktion der Schaltung überprüft:

```

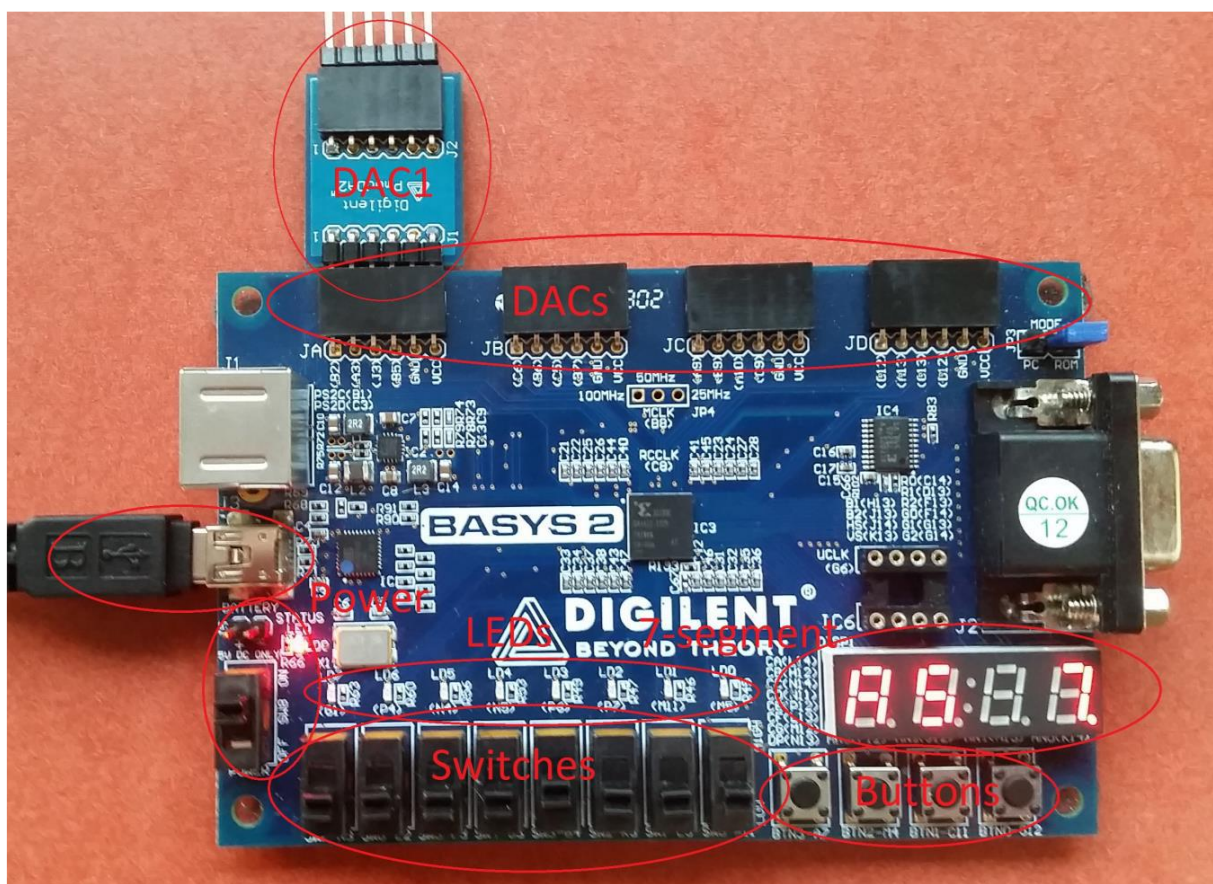
61  clk_process :process
62  begin
63      clk <= '0';
64      wait for clk_period/2;
65      clk <= '1';
66      wait for clk_period/2;
67  end process;
68
69  tick_process :process
70  begin
71      wait for 5.5 ns;
72      Schleife: loop
73          s_tick <= '0';
74          wait for clk_period*5;
75          s_tick <= '1';
76          wait for clk_period;
77      end loop;
78  end process;
79
80
81  rst<='1', '0' after 50 ns;
82  -- *** Test Bench - User Defined Section ***
83  tb : PROCESS
84  BEGIN
85      din <= "10010110";
86      wait for 85.5 ns;
87      tx_start <= '1';
88
89      wait for clk_period;
90      tx_start <= '0';
91      WAIT; -- will wait forever
92  END PROCESS;
93  -- *** End Test Bench - User Defined Section ***
94
95  END;
```

Die entsprechende Simulation sah folgendermaßen aus:



Man kann erkennen dass nach 8us der Simulation die gesendeten Daten erfolgreich empfangen werden.

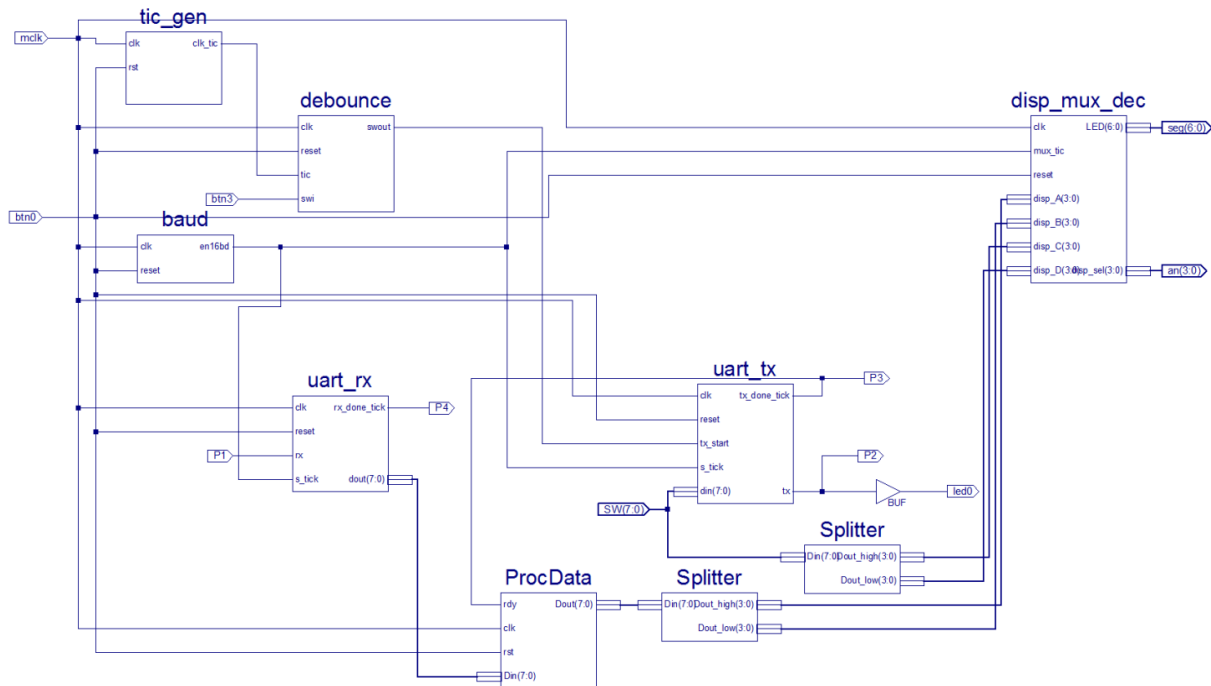
BASYS-Board:



Übung 1

Schematic

Um gewünschte Funktion zu erfüllen, musste das bereits bestehende Projekt erweitert werden.



Funktion

In diesem Fall wurde Button 0 als Reset-Signal verwendet. Um allerdings nur einen Tick an den „debounce“-Block zu senden, wenn der Knopf betätigt wurde, wurde der Block „tic_gen“ verwendet.

Um den Vorgang zu starten, musste Button 3 betätigt werden. Dies wurde umgesetzt indem dieser zuerst mit dem Block „debounce“ entprellt wurde und anschließend als Startsignal an den Sendeblock „uart_tx“ weitergegeben wurde.

Um die Übertragung später mit dem Oszilloskop messen zu können, wurde die Verbindung zwischen tx und rx mit einer Drahtbrücke realisiert.

Außerdem wurde LED 0 verwendet um die gesendeten Daten als kurzen Impuls aufleuchten zu lassen.

Die gesendeten und empfangenen Daten wurden jeweils mit dem Block „Splitter“ in obere und untere bits geteilt um diese anschließend über den Display-MUX an der 7-Segment-Anzeige ausgeben zu können.

Allerdings mussten die empfangenen Daten noch verarbeitet werden bevor diese ausgegeben werden konnten.

Dafür wurde die Entity „ProcData“ erstellt, sie inkrementiert die Daten und leitet sie anschließend weiter:

```
40 architecture Behavioral of ProcData is
41
42     signal help_in, help_out: unsigned(7)
43
44 begin
45
46     help_in <= unsigned(Din);
47
48     proc: process(rst, clk, help_in, rdy)
49     begin
50         if(rst = '1') then
51             help_out <= (others => '0');
52         else
53             if(clk'event and clk='1') then
54                 if(rdy = '1') then
55                     help_out <= help_in + 1;
56                 end if;
57             end if;
58         end if;
59     end process;
60
61     Dout <= std_logic_vector(help_out);
62
63 end Behavioral;
```

Um nun das Programm in das BASYS-Board zu spielen, musste das User Constraint File entsprechend konfiguriert werden.

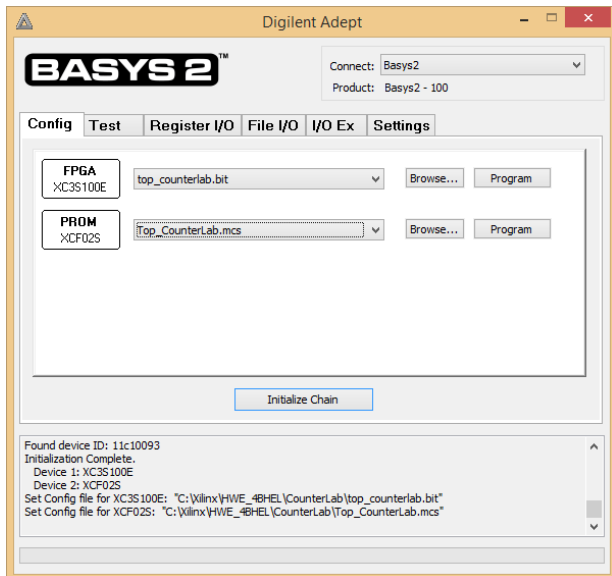
Für die Übertragung wurde eine Baud-Rate von 9600 verwendet, was in dem Block „baud“ eingestellt wurde.

Um also ein Programming-File zu erstellen mussten in der Xilinx Oberfläche die Schritte „Synthesize – XST“ -> „Implement Design“ -> „Generate Programming File“ durchgeführt werden.

Davor musste allerdings noch mit Rechtsklick auf „Generate Programming File“ -> „Process Properties“ -> „Startup Options“ der FPGA Start-Up Clock auf „JTAG Clock“ umgestellt werden.

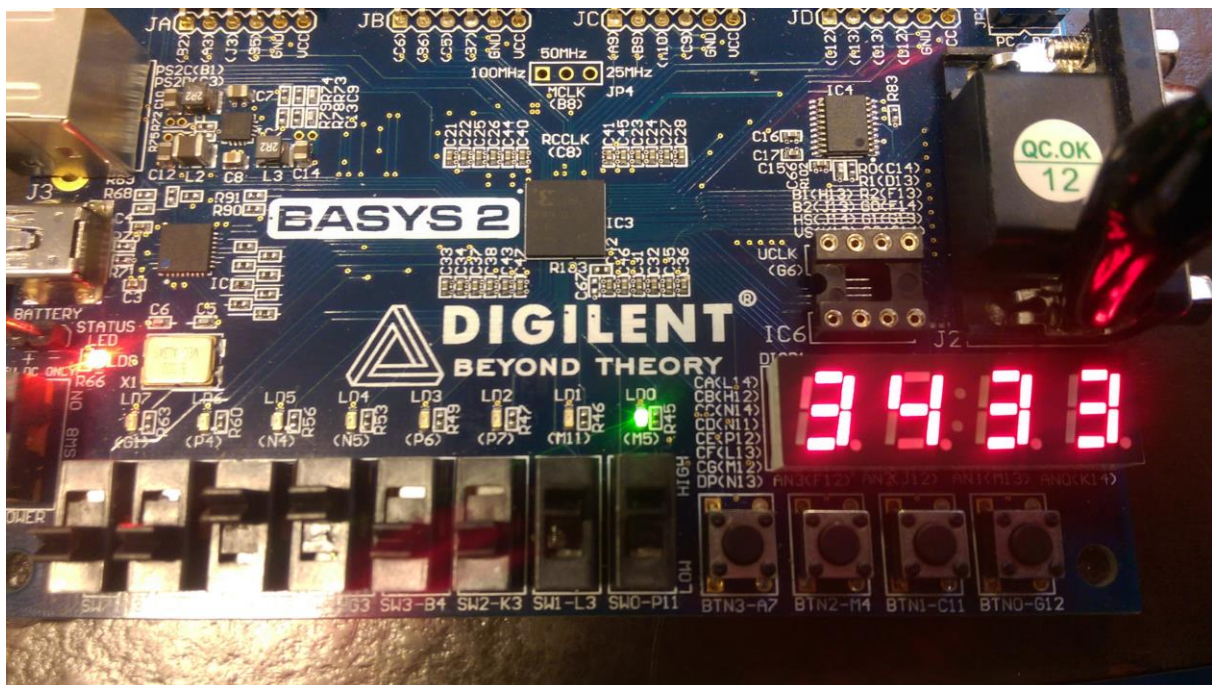
Funktionstest

Es wurde dann das Programm „Digilent Adept“ verwendet um das Programming-File in das Board zu laden:

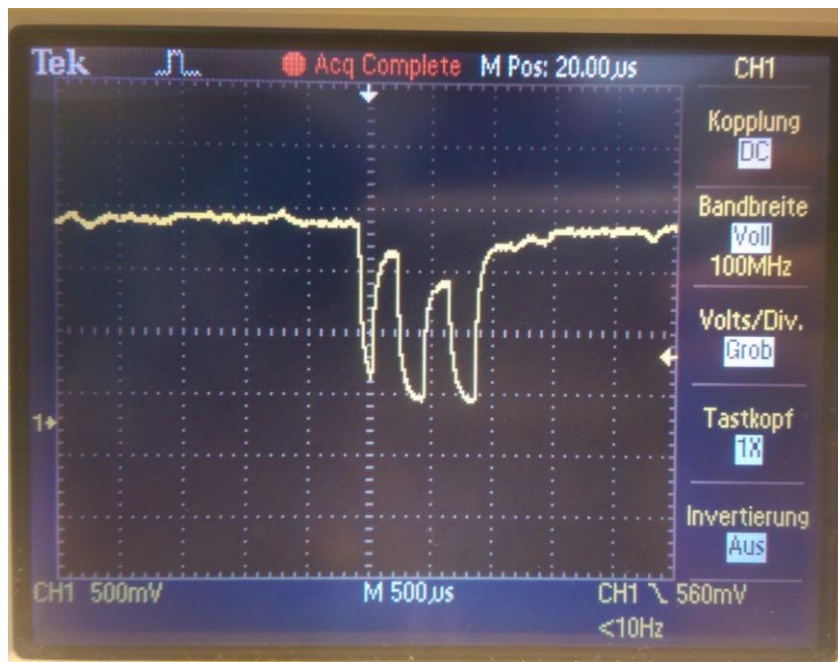


Da vergessen wurde während der Übung einen Screenshot aufzuzeichnen, stammt dieser aus einer vorherigen Laborübung.

Anschließend konnte festgestellt werden, dass die Funktion nach Wunsch durchgeführt wurde:



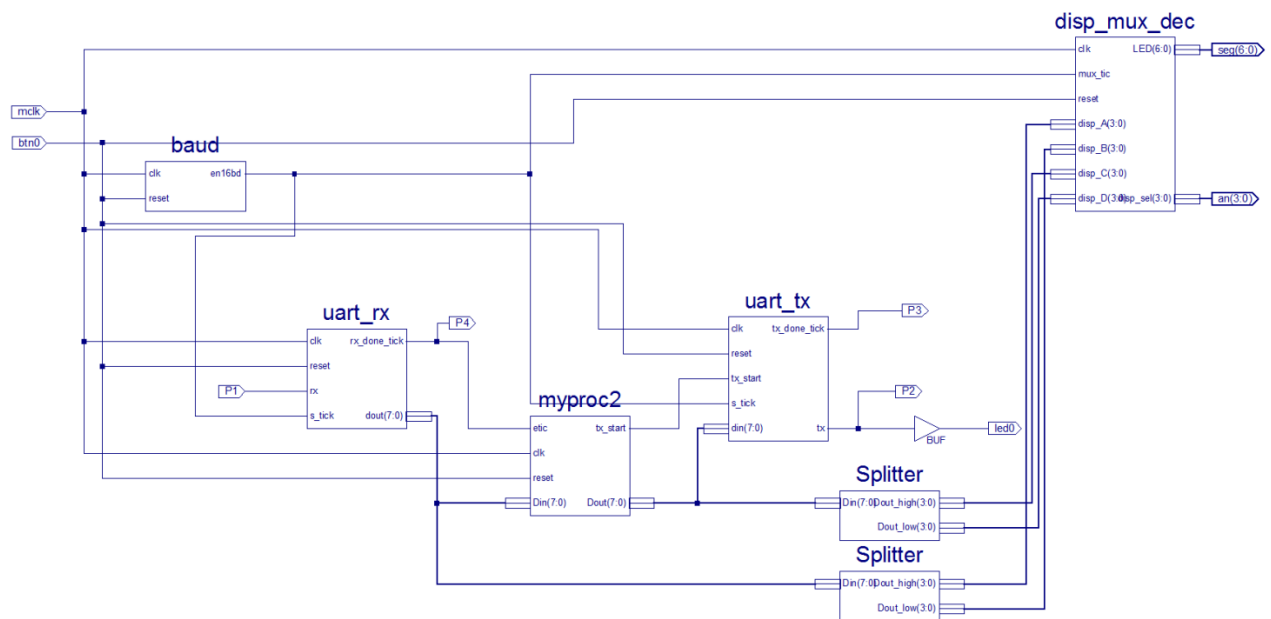
Die am Oszilloskop gemessene Datenübertragung sah folgender Maßen aus:



Übung 2

Schematic

Um gewünschte Erweiterung zu realisieren, wurde die Schematic folgender Maßen geändert:



Funktion

In dieser Schaltung existiert kein Start-Button mehr, denn die Daten werden sofort am PIN 1 empfangen.

Diese werden anschließend mit der aktualisierten Version „myproc2“ verarbeitet und über die 7-Segment Anzeige ausgegeben:

```
42 architecture Behavioral of myproc2 is
43
44 signal help: unsigned(7 downto 0);
45
46 begin
47
48   proc:process(Din,etic,clk, reset)
49   begin
50
51     if (reset = '1') then
52
53       help <= "00000000";
54       tx_start <= '0';
55
56
57     elsif(clk'event and clk = '1') then
58
59       if (etic = '1') then
60
61         help <= unsigned(Din)+1;
62         tx_start <= '1';
63
64       else
65
66         tx_start <= '0';
67
68       end if;
69     end if;
70
71   end process;
72
73   Dout <= std_logic_vector(help);
74
75 end Behavioral;
```

Sobald der RX-Block mit dem Empfang fertig ist, startet der der Block „myproc2“ mit dem inkrementieren der Daten und sendet er einen Start-Tic an den TX-Block der anschließend die verarbeiteten Daten über den PIN 2 weitersendet.

Zusätzlich werden die verarbeiteten Daten wieder in obere und untere bits geteilt und über den Display-Multiplexer ausgegeben.

Funktionstest

Nachdem ein Programming-File erstellt und in das BASYS-Board geladen wurde, wurde festgestellt dass ein Fehler in der Funktion bestand.

Nach der Feststellung, dass der verwendete Tastkopfdefekt war wurde dieser ausgetauscht, doch dies behob den Fehler in der Funktion nicht.

Anschließend wurde der Übungsleiter zu Hilfe gerufen, doch auch dieser konnte den Fehler nicht finden. Es wurde vermutet dass der ESD-Schutz des PINs defekt war.

Erkenntnis

In dieser Übung wurden Kenntnisse in VHDL aufgefrischt und erweitert.

Außerdem wurde die Projektentwicklung in Xilinx und die Fehlersuche am BASYS-Board und in Xilinx trainiert.

Zusätzlich wurde Wissen bezüglich IP-Cores und deren Verwendung angeeignet.

Darüber hinaus wurde der UART wiederholt.